DATALAD

DECENTRALIZED MANAGEMENT OF DIGITAL OBJECTS FOR OPEN SCIENCE

Adina Wagner
mas.to/@adswa
Psychoinformatics lab,
Institute of Neuroscience and Medicine, Brain & Behavior (INM-7)
Research Center Jülich

Slides: DOI 10.5281/zenodo.10556597 (Scan the QR code)
files.inm7.de/adina/talks/html/hamburg_2024.html

ACKNOWLEDGEMENTS

Funders

NSF 1429999

germany-usa

SPONSORED BY THE
Federal Ministry of Education and Research
BMBF 01GQ1411

JÜLICH Forschungszentrum

DFG

EUROPEAN UNION
European Regional Development Fund

cbbs
center for behavioral brain sciences

SACHSEN-ANHALT
Ministerium für Wissenschaft und Wirtschaft

DataLad software & ecosystem

- Psychoinformatics Lab, Research center Jülich
- Center for Open Neuroscience, Dartmouth College
- Joey Hess (git-annex)
- *>100 additional contributors*

DataLad Office Hour

Every Tuesday, 4pm.
Join the Matrix Chatroom!

Collaborators

Human Brain Project

CONP PCNO

OpenNEURO

eBRAIN Health

MOTOR SFB 1451

brainlife.io

cbrain

VirtualBrainCloud

IMPROVE SCIENTIFIC WORKFLOWS, COMING FROM THE PERSPECTIVE OF SOFTWARE DISTRIBUTIONS AND DEVELOPMENT

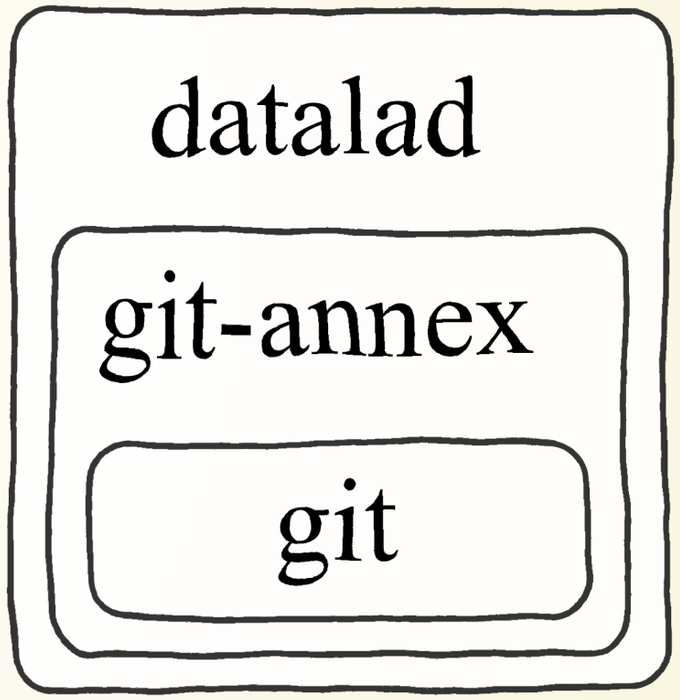
DataLad

"Share and treat data like software"

NeuroDebian

git

DATALAD DATASETS



A DataLad dataset is a joined Git + git-annex repository

WHAT MAKES SCIENTIFIC WORKFLOWS SPECIAL?

Scientific building blocks are not static.

The building blocks of a scientific result are rarely static

Analysis code, manuscripts, ... evolve

(Rewrite, fix bugs, add functions, refactor, extend, ...)

Image credit: Based on Piled Higher and Deeper 1531

Image credit: Scriberia and The Turing Way (CC-BY)

VERSION CONTROL

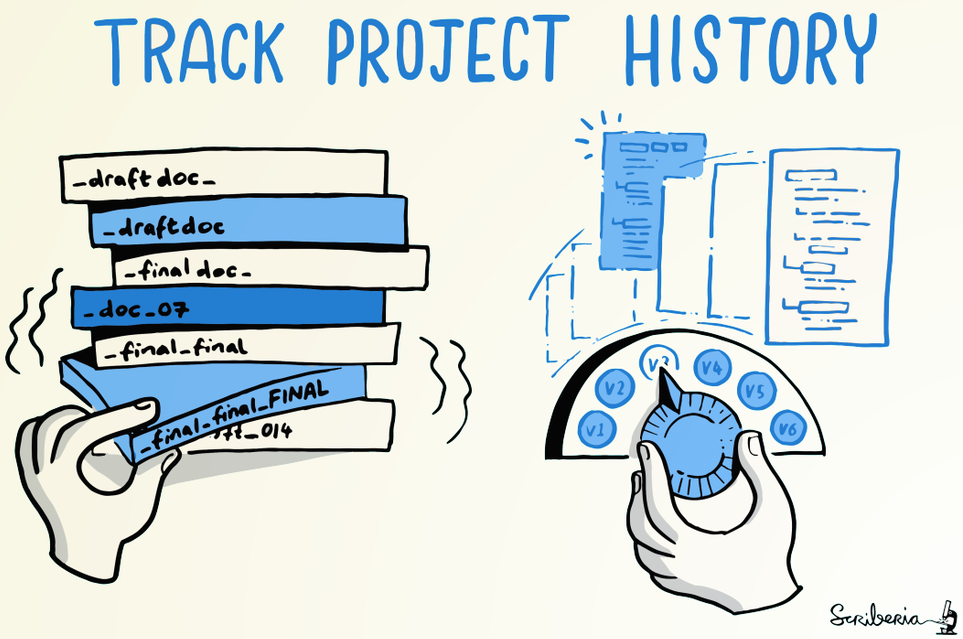


Image credit: CC-BY Scriberia & The Turing Way

- keep things organized
- keep track of changes
- revert changes or go back to previous states
- collect and share digital provenance
- industry standard: Git


```
2022-01-30 15:47 +0100 Michael Hanke      o Be explicit re FAIRification
2022-01-30 15:27 +0100 Michael Hanke      o Add statement on numerical precision
2022-01-30 11:36 +0100 Michael Hanke      o (Re)define RIA
2022-01-30 11:04 +0100 Małgorzata Wierzba o Add MW's funding
2022-01-28 17:05 +0100 Felix Hoffstaedter o reword bitidentity comment on reproducebility
2022-01-28 16:33 +0100 Adina Wagner       o Remove 'powerful' from snakemake's description as it is unspecific
2022-01-28 16:07 +0100 Adina Wagner       o R1: Finish the sentences on Dask and Spark
2022-01-28 15:10 +0100 Adina Wagner       o Revert "Move reference to {fig:imageqc} to results as well"
2022-01-28 14:35 +0100 Adina Wagner       o Add the compiled bibliography file into the repo, needed in resubmission
2022-01-28 14:28 +0100 Adina Wagner       o Apply @loj's suggestion on Parsl
2022-01-28 12:12 +0100 Małgorzata Wierzba o Minor tweak
2022-01-28 11:40 +0100 Małgorzata Wierzba o Fix typo
2022-01-28 11:36 +0100 Małgorzata Wierzba o Move reference to {fig:imageqc} to results as well
2022-01-28 10:11 +0100 Małgorzata Wierzba o Minor tweak
```

The building blocks of a scientific result are rarely static

Data changes, too

(errors are fixed, data is extended, naming standards change, an analysis
requires only a subset of your data...)

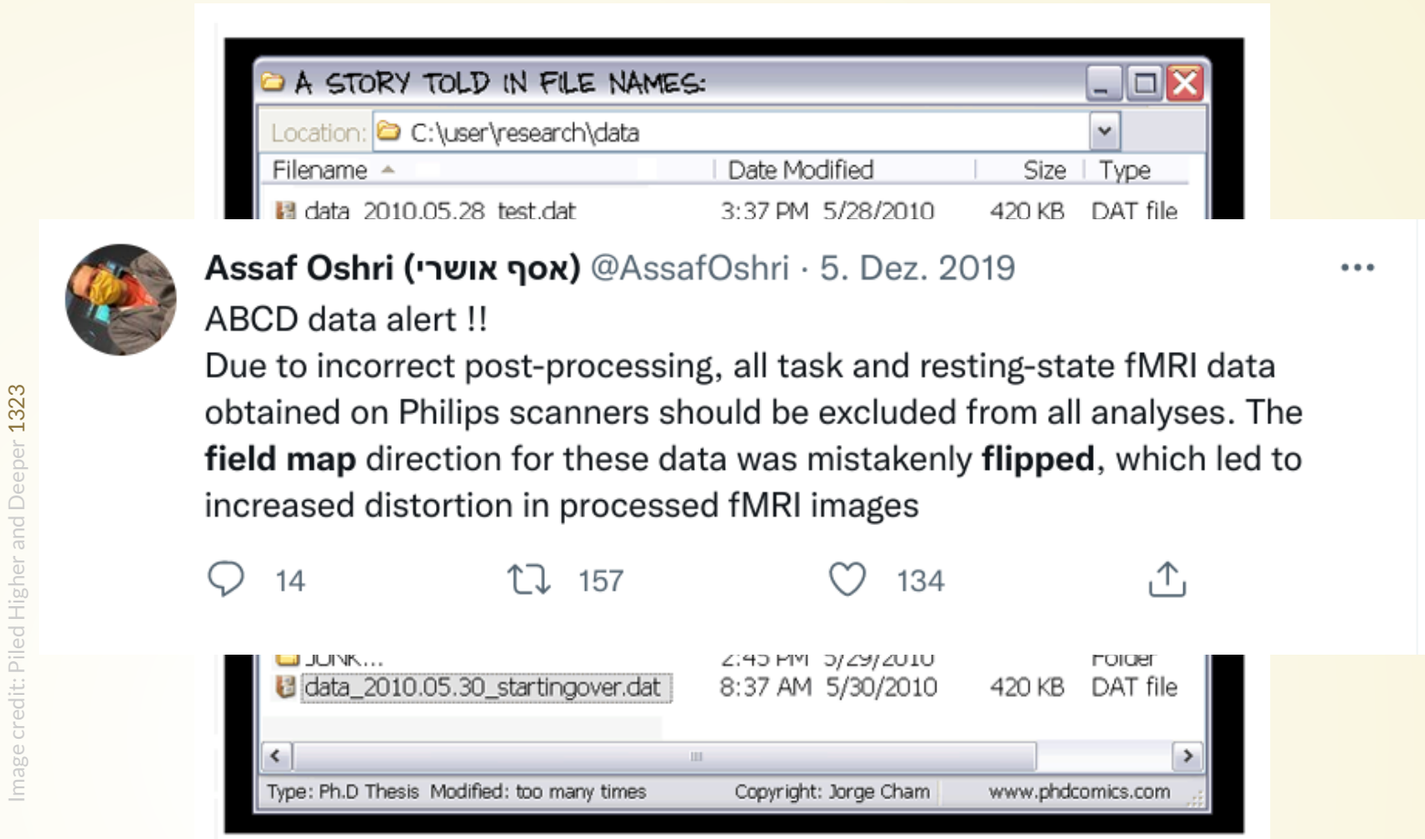
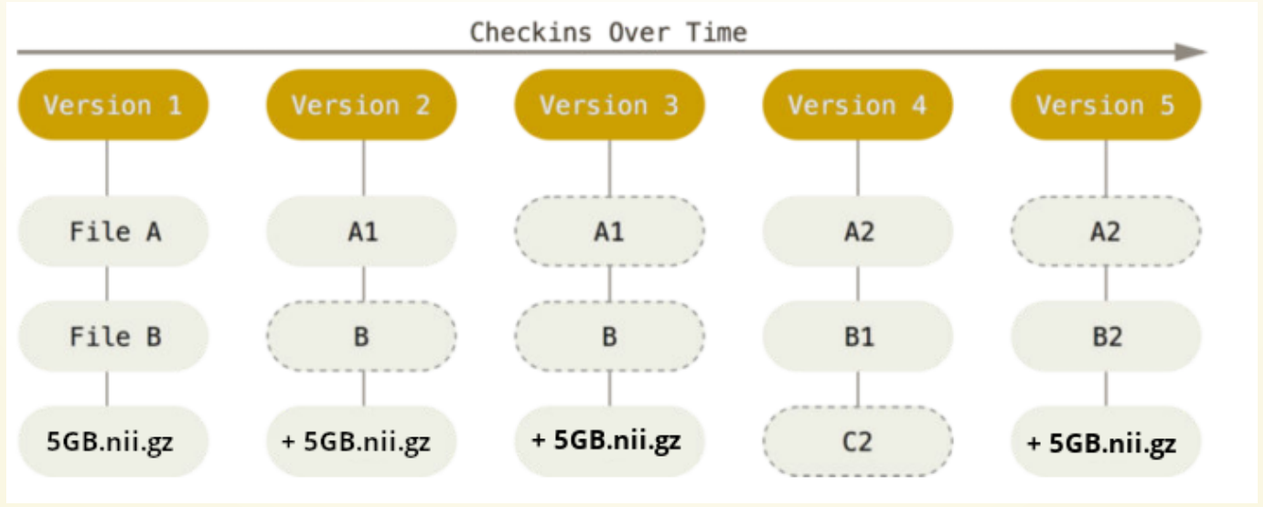


Image credit: Piled Higher and Deeper 1323

Sadly, Git does not handle large files well.



VERSION CONTROL BEYOND TEXT FILES

Using git-annex, DataLad version controls large data

```
2020-03-13 10:46 +0100 Adina Wagner     o [DATALAD RUNCMD] add non-defaced
2020-03-13 10:29 +0100 Adina Wagner     o [DATALAD RUNCMD] reconvert DICOM
2018-05-11 09:23 +0200 Michael Hanke    o [master] {origin/HEAD} {origin/m
2018-05-11 09:19 +0200 Michael Hanke    o Enable DataLad metadata extracto
2018-05-11 09:17 +0200 Michael Hanke    o [DATALAD] new dataset
2018-05-11 09:17 +0200 Michael Hanke    o [DATALAD] Set default backend fo
2018-01-19 14:19 +0100 Michael Hanke    o <v1.5> Update changelog for 1.5
2018-01-19 14:09 +0100 Michael Hanke    o BF: Re-import respiratory trace
2018-01-14 18:59 +0100 Michael Hanke    o Fix type in physio log converter
2017-01-10 10:10 +0100 Michael Hanke    o ENH: Report per-stimulus events
2016-12-10 20:18 +0100 Michael Hanke    o Add BIDS-compatible stimuli/ dir
2016-11-15 07:04 +0100 Michael Hanke    o Minor tweaks to gaze overlay scr
2016-10-30 11:03 +0100 Michael Hanke    o Add "TaskName" meta data field f
2016-09-21 08:33 +0200 Michael Hanke    o Add task-*_physio.json files
2016-09-21 08:23 +0200 Michael Hanke    o BF: Fix task label in file names
2016-08-04 13:14 +0200 Michael Hanke    o Update changelog
2016-08-03 22:22 +0200 Michael Hanke    o Add cut position information to
2016-05-27 17:35 +0200 Michael Hanke    o {origin/_} Mention openfmri as d
2016-04-04 09:31 +0200 Michael Hanke    o Update publication links
2016-03-31 11:26 +0200 Michael Hanke    o Disable invalid test
[main] 6da25fb6fee2c698d35f52066698b6f94850f4d2 - commit 10 of 79  27%
```

```
commit 6da25fb6fee2c698d35f52066698b6f94850f4d2
Refs: v1.0-19-g6da25fb6
Author:     Michael Hanke <michael.hanke@gmail.com>
AuthorDate: Fri Jan 19 14:09:53 2018 +0100
Commit:     Michael Hanke <michael.hanke@gmail.com>
CommitDate: Fri Jan 19 14:11:23 2018 +0100

    BF: Re-import respiratory trace after bug fix in converter (fixes gh-
---
 ...er_task-movielocalizer_run-1_recording-cardresp_physio.tsv.gz | 2 +-
 ..._task-objectcategories_run-1_recording-cardresp_physio.tsv.gz | 2 +-
 ..._task-objectcategories_run-2_recording-cardresp_physio.tsv.gz | 2 +-
 ..._task-objectcategories_run-3_recording-cardresp_physio.tsv.gz | 2 +-
 ..._task-objectcategories_run-4_recording-cardresp_physio.tsv.gz | 2 +-
 ...calizer_task-retmapccw_run-1_recording-cardresp_physio.tsv.gz | 2 +-
 ...calizer_task-retmapclw_run-1_recording-cardresp_physio.tsv.gz | 2 +-
 ...calizer_task-retmapcon_run-1_recording-cardresp_physio.tsv.gz | 2 +-
 ...calizer_task-retmapexp_run-1_recording-cardresp_physio.tsv.gz | 2 +-
 ...2_ses-movie_task-movie_run-1_recording-cardresp_physio.tsv.gz | 2 +-
 ...2_ses-movie_task-movie_run-2_recording-cardresp_physio.tsv.gz | 2 +-
[diff] 6da25fb6fee2c698d35f52066698b6f94850f4d2 - line 1 of 2391  0%
```

VERSION CONTROL BEYOND TEXT FILES

- Datasets can have an optional **annex** for tracking (large) files without placing their content into Git
- For **annex'ed** files, identity (hash) and location information is put into Git, rather than their content:
  - Where the filesystem allows it, annexed files are symlinks:

```
$ ls -l sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz                    copy
lrwxrwxrwx 1 adina adina 142 Jul 22 19:45 sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz ->
../../.git/annex/objects/kZ/K5/MD5E-s24180157--aeb0e5f2e2d5fe4ade97117a8cc5232f.nii.gz/MD5E-s241
--aeb0e5f2e2d5fe4ade97117a8cc5232f.nii.gz
```

(PS: especially useful in datasets with many identical files)

  - The symlink reveals this internal data organization based on identity hash:

```
$ md5sum sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz                   copy
aeb0e5f2e2d5fe4ade97117a8cc5232f  sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz
```

  - The (tiny) symlink instead of the (potentially large) file content is committed - version controlling precise file identity without checking contents into Git

```
diff --git a/sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz b/sub-02/func/sub-02_task-oneback_run-01_bold.nii.
new file mode 120000
index 0000000..398e7f1
--- /dev/null
+++ b/sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz
@@ -0,0 +1 @@
+../../.git/annex/objects/kZ/K5/MD5E-s24180157--aeb0e5f2e2d5fe4ade97117a8cc5232f.nii.gz/MD5E-s24180157--aeb0e5f2e2
```

VERSION CONTROL BEYOND TEXT FILES

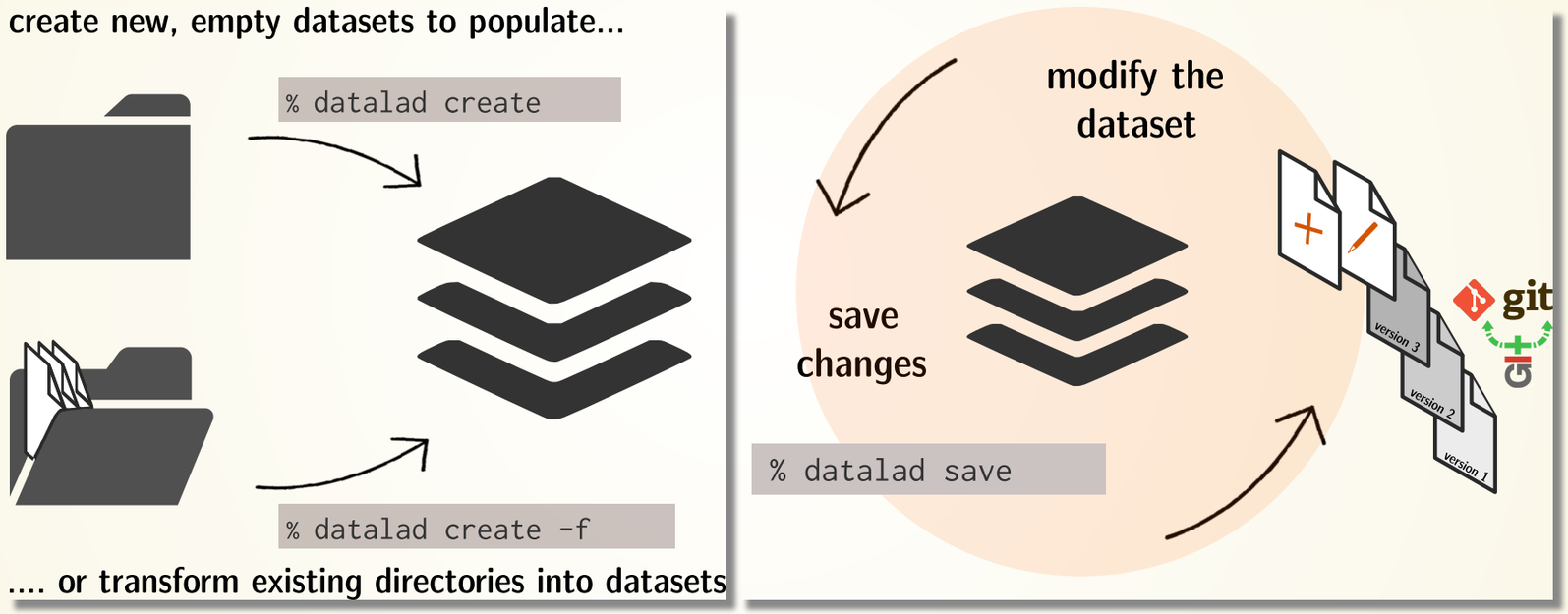
- Datasets can have an optional **annex** for tracking (large) files without placing their content into Git
- For **annex'ed** files, identity (hash) and location information is put into Git, rather than their content:
  - File availability information is stored to record a decentral network of file content. A file can exist in multiple different locations.

```
$ git annex whereis sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz
whereis sub-02/func/sub-02_task-oneback_run-01_bold.nii.gz (2 copies)
    8c3680dd-6165-4749-adaa-c742232bc317 -- git@8242caf9acd8:/data/repos/adswa/bidsdata.gi
    fff8fdbc-3185-4b78-bd12-718717588442 -- adina@muninn:~/bids-data [here]
ok
```

Delineation and advantages of decentral versus central RDM: Hanke et al., (2021). In defense of decentralized research data management

VERSION CONTROL

- DataLad knows two things: Datasets and files



- Every file you put into a in a dataset can be easily version-controlled, regardless of size, with the same command: `datalad save`

VERSION CONTROL

- Example: Add a new file into a dataset

```
# create a data analysis script
$ datalad status
untracked: code/script.py (file)
$ git status
On branch master
Untracked files:
  (use "git add file..." to include in what will be committed)
  code/script.py

nothing added to commit but untracked files present (use "git add" to
```

- Save the dataset modification...
  - ... with DataLad

```
$ datalad save \
    -m "Add a k-nearest-neighbour clustering analysis" \
    code/script.py
```

  - ... versus with Git

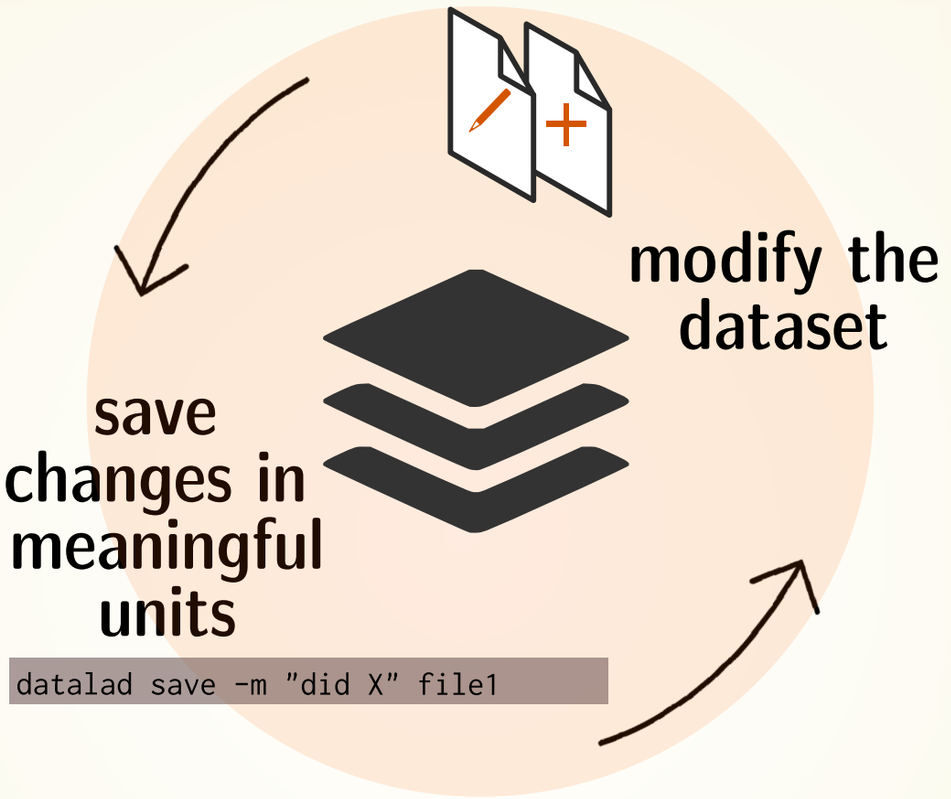
```
$ git add code/script.py
$ git commit -m "Add a k-nearest-neighbour clustering analysis"
```

  - ... versus with git-annex

```
$ git annex add code/script.py
$ git commit -m "Add a k-nearest-neighbour clustering analysis"
```

LOCAL VERSION CONTROL

Procedurally, version control is easy with DataLad!



Stay flexible:

- Non-complex DataLad core API (easier than Git)
- Pure Git or git-annex commands (for regular Git or git-annex users, or to use specific functionality)

Advice:

- Save meaningful units of change
- Attach helpful commit messages

VERSION CONTROL REGARDLESS OF SIZE

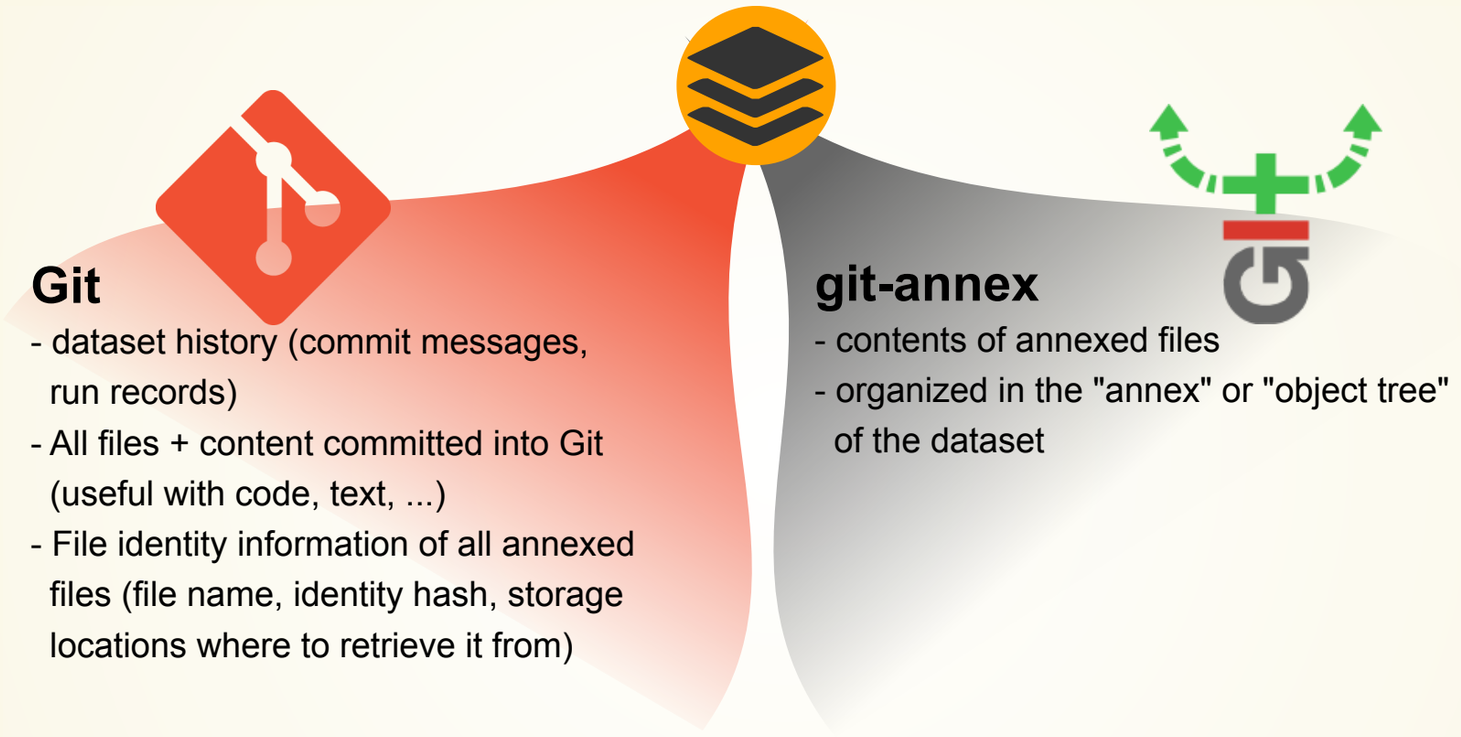
```
$ datalad save \
  -m "Adding raw data from neuroimaging study 1" \
  sub-*
  add(ok): sub-1/anat/T1w.json (file)
  add(ok): sub-1/anat/T1w.nii.gz (file)
  add(ok): sub-1/anat/T2w.json (file)
  add(ok): sub-1/anat/T2w.nii.gz (file)
  add(ok): sub-1/func/sub-1-run-1_bold.json (file)
  add(ok): sub-1/func/sub-1-run-1_bold.nii.gz (file)
  add(ok): sub-10/anat/T1w.json (file)
  add(ok): sub-10/anat/T1w.nii.gz (file)
  add(ok): sub-10/anat/T2w.json (file)
  add(ok): sub-10/anat/T2w.nii.gz (file)
[110 similar messages have been suppressed]
  save(ok): . (dataset)
  action summary:
add (ok: 120)
save (ok: 1)
```

copy

GIT VERSUS GIT-ANNEX

Data in datasets is either stored in Git or git-annex

By default, everything is annexed, i.e., stored in a dataset annex



Git	git-annex
handles **small** files well (text, code)	handles **all** types and sizes of files well
file contents are in the Git history and will be **shared** upon git/datalad push	file contents are in the annex. Not necessarily shared
Shared with every dataset clone	**Can be kept private** on a per-file level when sharing the dataset
Useful: Small, non-binary, frequently modified, need-to-be-accessible (DUA, README) files	Useful: Large files, private files

FROM HERE

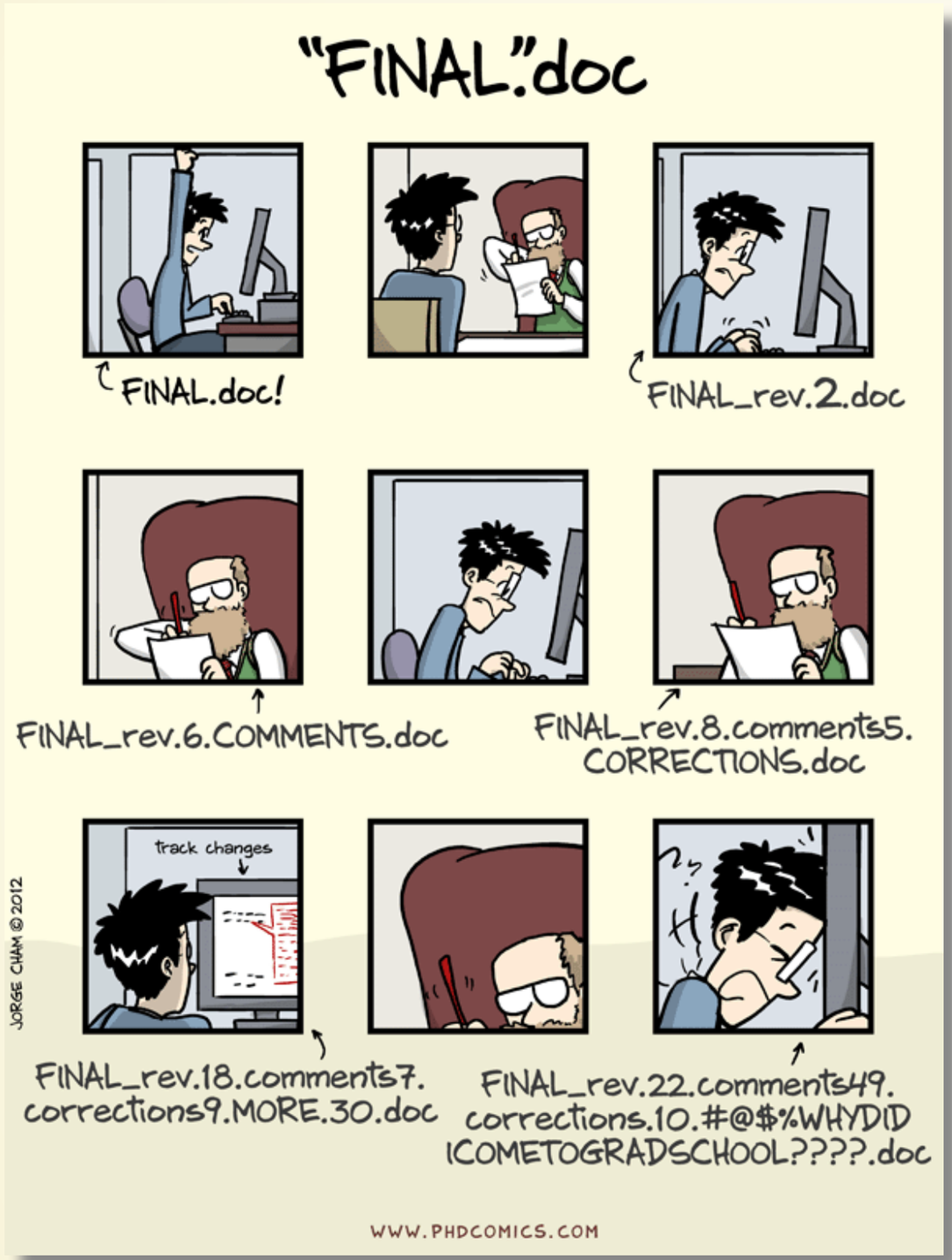


Image credit: www.phdcomics.com; www.linode.com

TO THIS:

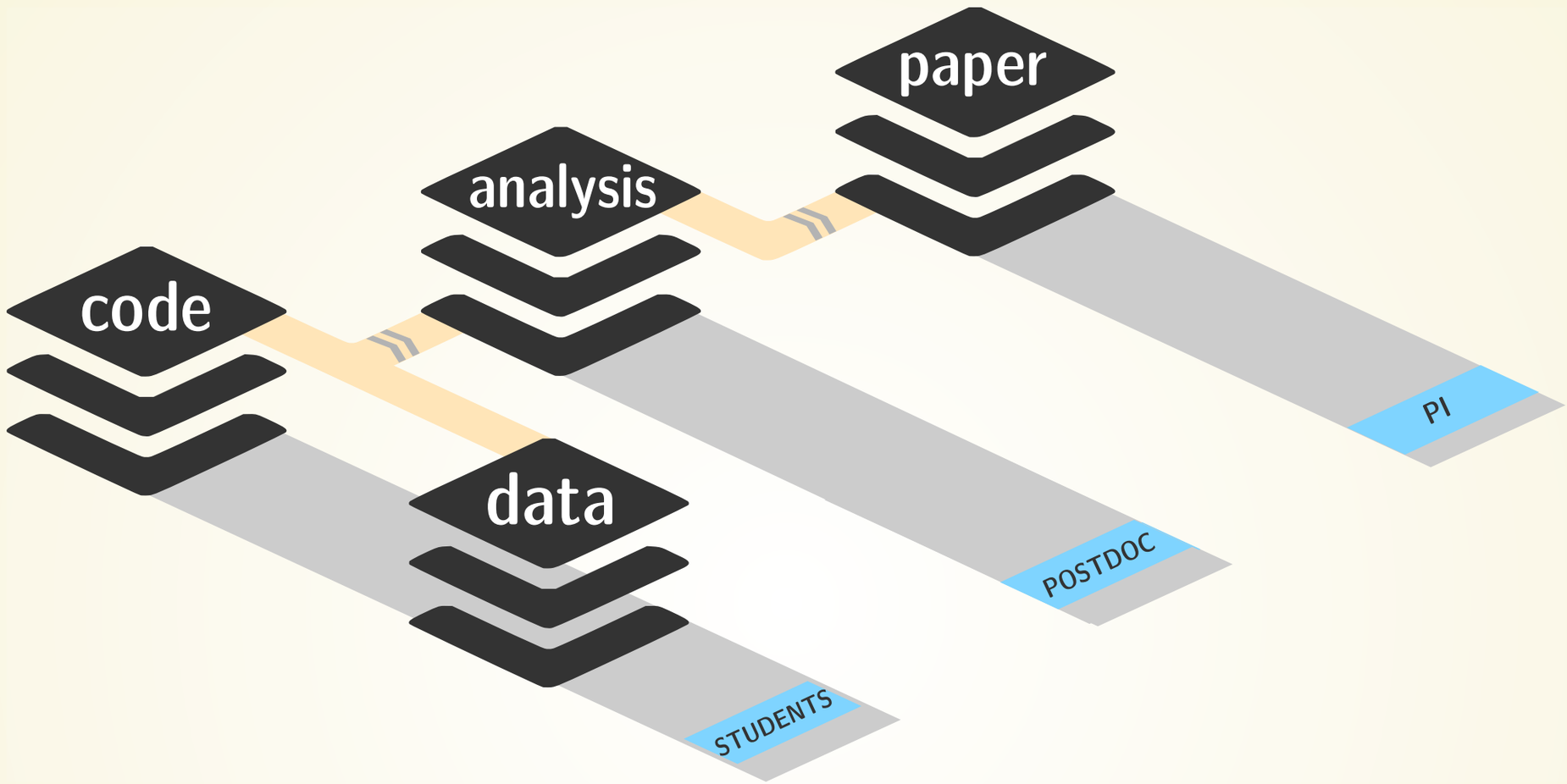
BUT: Version control is only one aspect of data management

WHAT MAKES SCIENTIFIC WORKFLOWS SPECIAL?

Scientific building blocks are not static.

Version control beyond text

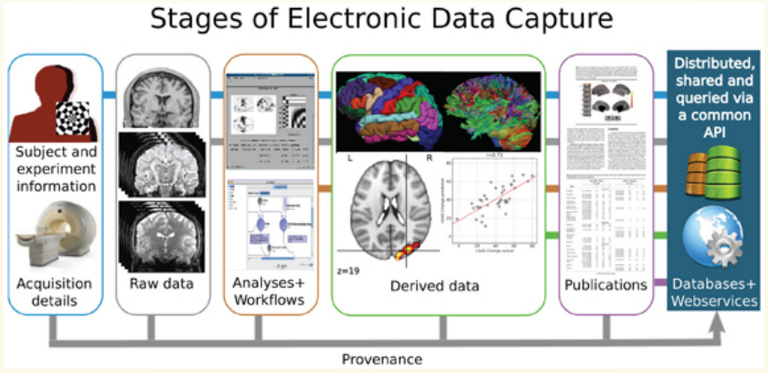
Science is build from modular units.



- Typical workflow in science
  - Prior works (algorithm development, empirical data, etc.) are combined to produce novel results with to goal of a publication
  - **Aggregation across time and contributors**
  - Aiming for (but often failing) to be reproducible

VERSION CONTROL BEYOND SINGLE REPOSITORIES

- **Why** are multiple repositories needed (in science)?
  - Size impacts I/O and logistics
    - Git can struggle with 1M+ files or 100k+ commits
    - Filesystems (licensing) can struggle with large numbers of inodes
  - Target audience is different
    - Public vs. private or personal vs. anonymized data
  - Pace of evolution or access patterns are different
    - "Factual" raw data vs. choices of (pre-)processing
    - Completed acquisition vs. ongoing study



GIT SUBMODULES

- Built-in Git feature: Add a repository to another repository, treating them as separate projects (e.g., use third party project, but keep commits separate)

Make a project with a submodule:

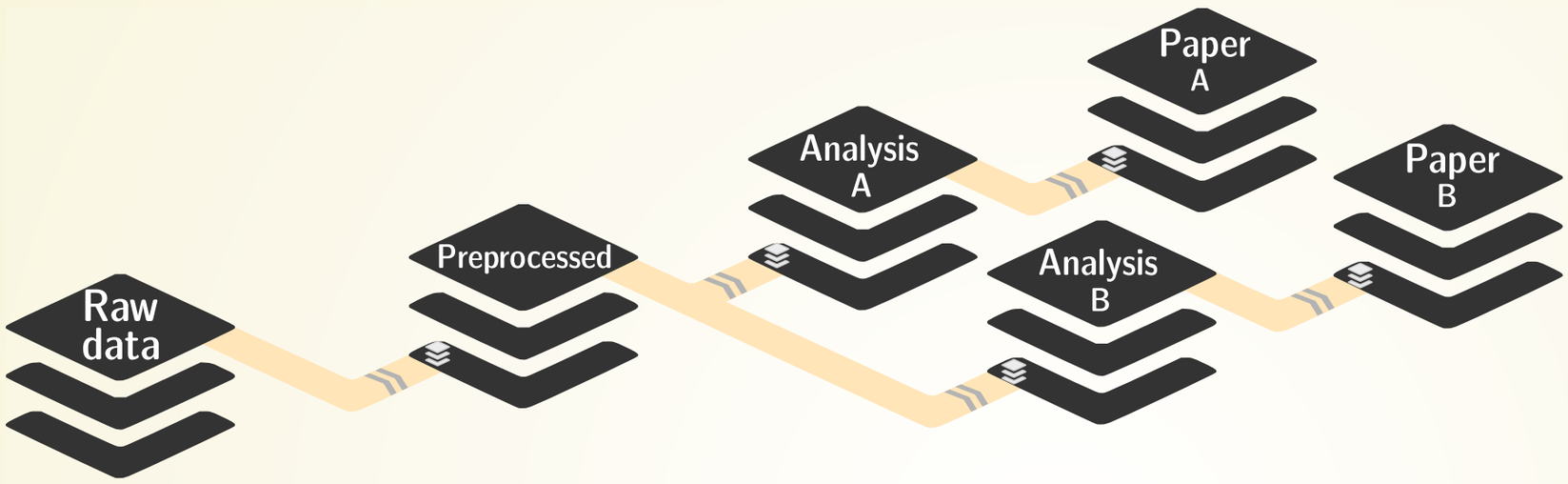
```
$ git init myproject
Initialized empty Git repository in /tmp/myproject/
$ cd myproject
$ git submodule add \
    https://github.com/adswa/multimatch_gaze.git
Cloning into '/tmp/myproject/multimatch_gaze'...
done.
$ git commit -am 'Add multimatch module'
[main fb9093c] Add multimatch module
 2 files changed, 4 insertions(+)
 create mode 100644 .gitmodules
 create mode 160000 multimatch_gaze
```

Get a repository with a submodule:

```
$ git clone https://github.com/adswa/mypr
Cloning into 'myproject'...
done.
$ cd myproject
$ git submodule init
Submodule 'multimatch_gaze' (https://gith
registered for path 'multimatch_gaze'
```

DATASET NESTING

- Seamless nesting mechanisms:



Nest modular datasets to create a linked hierarchy of datasets, and enable recursive operations throughout the hierarchy

  - hierarchies of datasets in super-/sub-dataset relationships
  - based on Git submodules, but more seamless
- Overcomes scaling issues with large amounts of files

```
adina@bulk1 in /ds/hcp/super on git:master❱ datalad status --annex -r
15530572 annex'd files (77.9 TB recorded total size)
nothing to save, working tree clean
```

(github.com/datalad-datasets/human-connectome-project-openaccess)

- Modularizes research components for transparency, reuse, and access management

KEEPING A PROJECT CLEAN AND ORDERLY

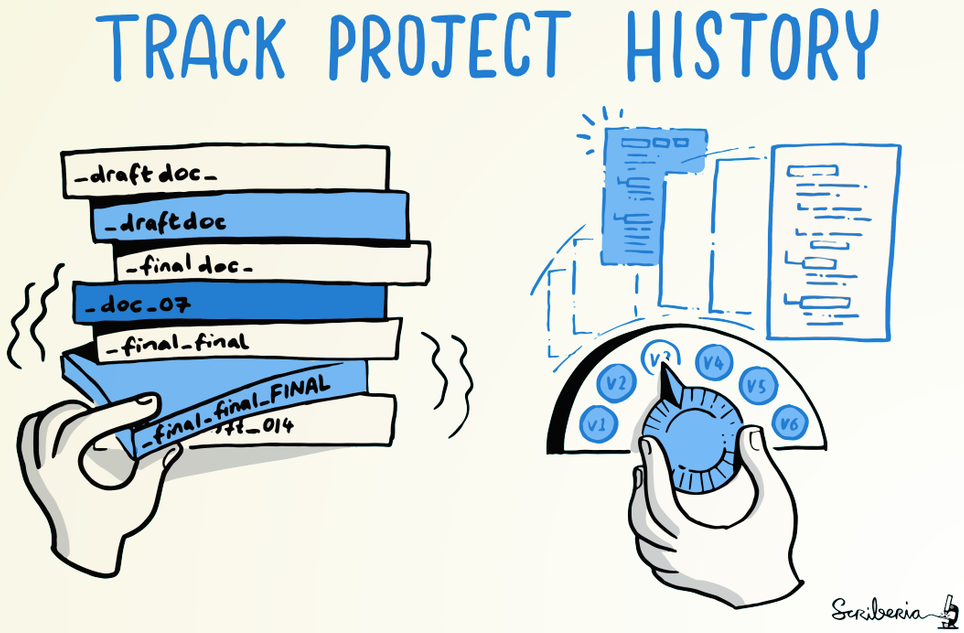
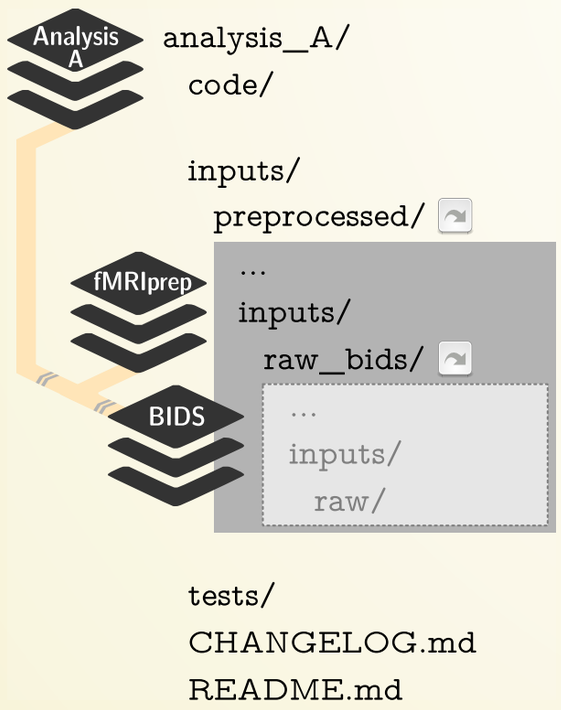


Image credit: CC-BY Scriberia & The Turing Way

Version control

- keep things organized
- keep track of changes
- revert changes or go back to previous states



Intuitive structure

- Keep projects lean
- Link project dependencies easily
- Follow the YODA principles

KEEPING A PROJECT CLEAN AND ORDERLY

First, let's create a new data analysis dataset with `datalad create`

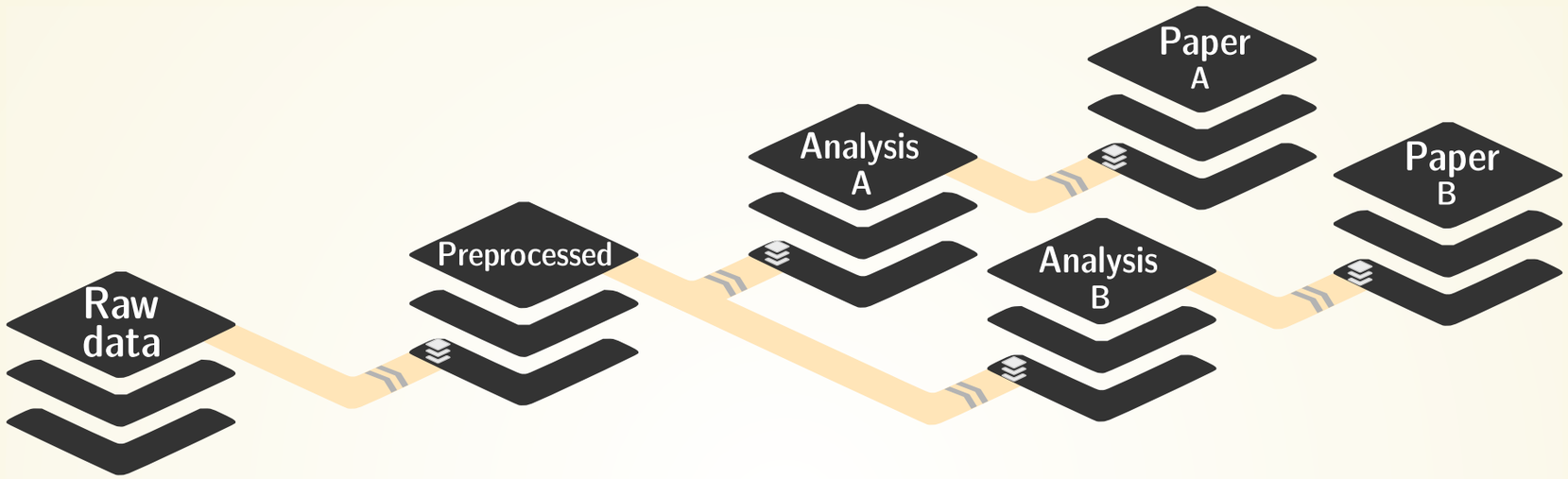
```
$ datalad create -c yoda myanalysis                                    copy
[INFO   ] Creating a new annex repo at /tmp/myanalysis
[INFO   ] Scanning for unlocked files (this may take some time)
[INFO   ] Running procedure cfg_yoda
[INFO   ] == Command start (output follows) =====
[INFO   ] == Command exit (modification check follows) =====
create(ok): /tmp/myanalysis (dataset)
```

- `-c yoda` applies useful pre-structuring and configurations:

```
$ tree                                                                 copy
.
├── CHANGELOG.md
├── code
│   └── README.md
└── README.md
```

INTUITIVE DATA ANALYSIS STRUCTURE

- You can link datasets together in superdataset-subdataset hierarchies:



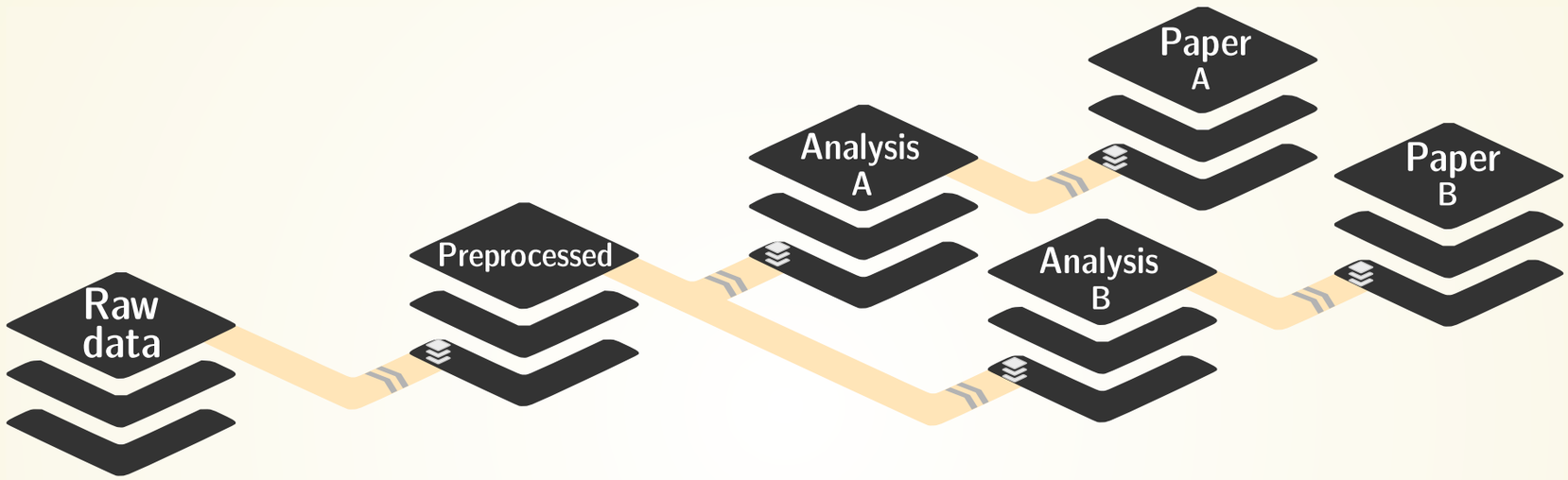
Nest modular datasets to create a linked hierarchy of datasets, and enable recursive operations throughout the hierarchy

```
$ cd myanalysis
# we can install analysis input data as a subdataset to the dataset
$ datalad clone -d . https://github.com/datalad-handbook/iris_data.git input/
[INFO   ] Scanning for unlocked files (this may take some time)
[INFO   ] Remote origin not usable by git-annex; setting annex-ignore
install(ok): input (dataset)
add(ok): input (file)
add(ok): .gitmodules (file)
save(ok): . (dataset)
action summary:
  add (ok: 2)
  install (ok: 1)
  save (ok: 1)
```

copy

INTUITIVE DATA ANALYSIS STRUCTURE

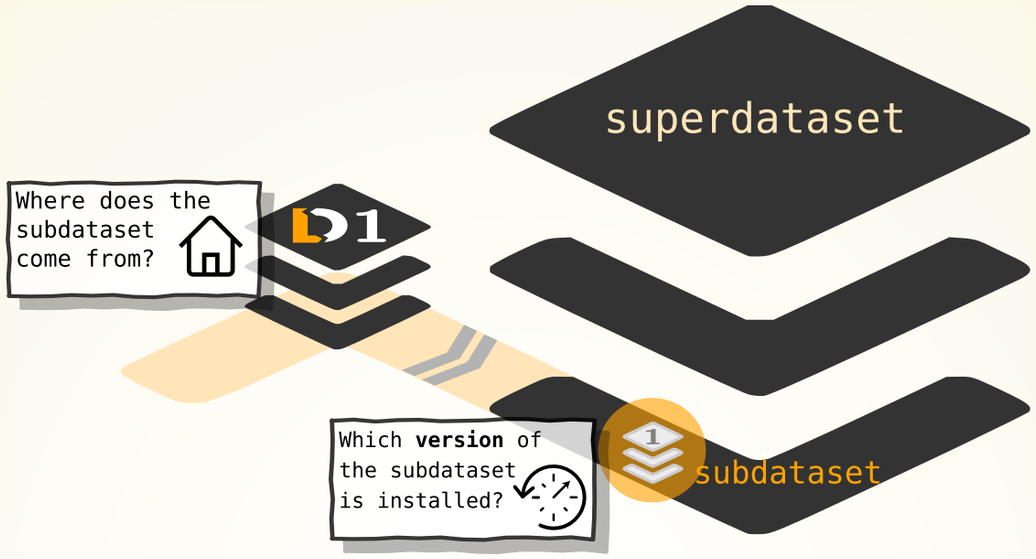
- You can link datasets together in superdataset-subdataset hierarchies:



Nest modular datasets to create a linked hierarchy of datasets, and enable recursive operations throughout the hierarchy

```
$ tree                                                    copy
.
├── CHANGELOG.md
├── code
│   ├── README.md
│   └── script.py
└── input
    └── iris.csv
```

SEAMLESS DATASET NESTING & LINKAGE




```
$ datalad clone --dataset . https://github.com/datalad-handbook/iris_data.git input/
```

copy

```
$ git diff HEAD~1
diff --git a/.gitmodules b/.gitmodules
new file mode 100644
index 0000000..c3370ba
--- /dev/null
+++ b/.gitmodules
@@ -0,0 +1,3 @@
+[submodule "input"]
+       path = input
+       datalad-id = 68bdb3f3-eafa-4a48-bddd-31e94e8b8242
+       datalad-url = https://github.com/datalad-handbook/iris_data.git
diff --git a/input b/input
new file mode 160000
index 0000000..fabf852
--- /dev/null
+++ b/input
@@ -0,0 +1 @@
```

copy

WHAT MAKES SCIENTIFIC WORKFLOWS SPECIAL?

Scientific building blocks are not static.

Version control beyond text

Science is build from modular units.

Nesting

Science is exploratory, iterative, multi-stepped, and complex.

REUSING PAST WORK ISN'T NECESSARILY SIMPLE

Your past self is the worst collaborator:

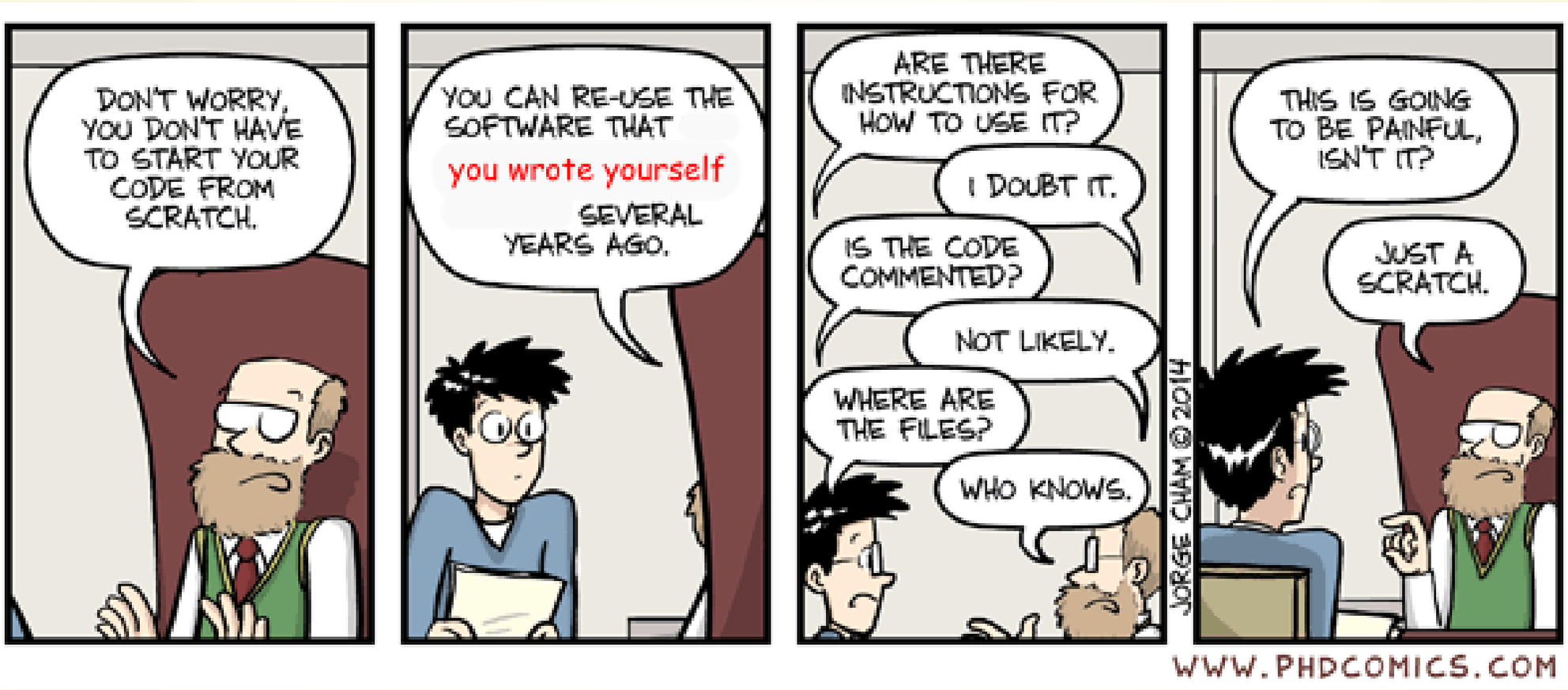


Image credit: Full comic at http://phdcomics.com/comics.php?f=1979

LEAVING A TRACE

"Shit, which version of which script produced these outputs from which version of what data?"

"Shit, why buttons did I click and in which order did I use all those tools?"

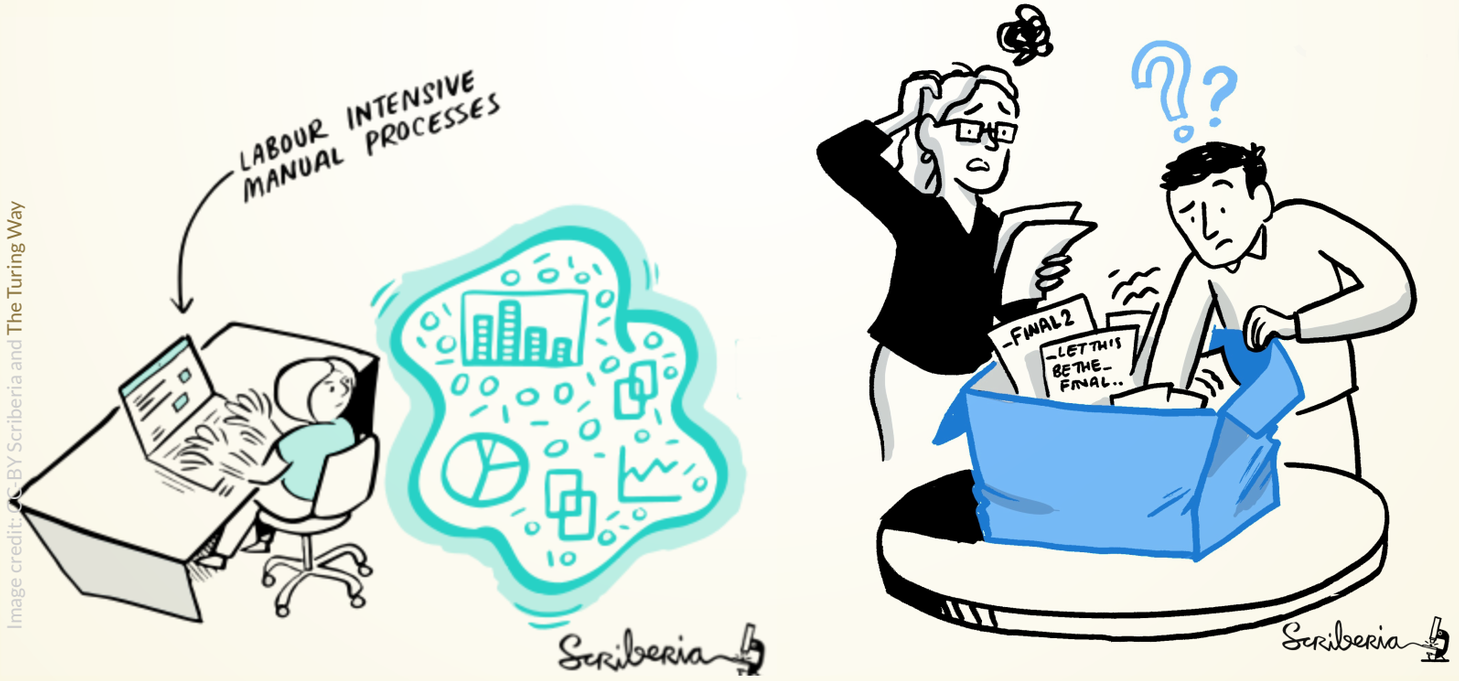


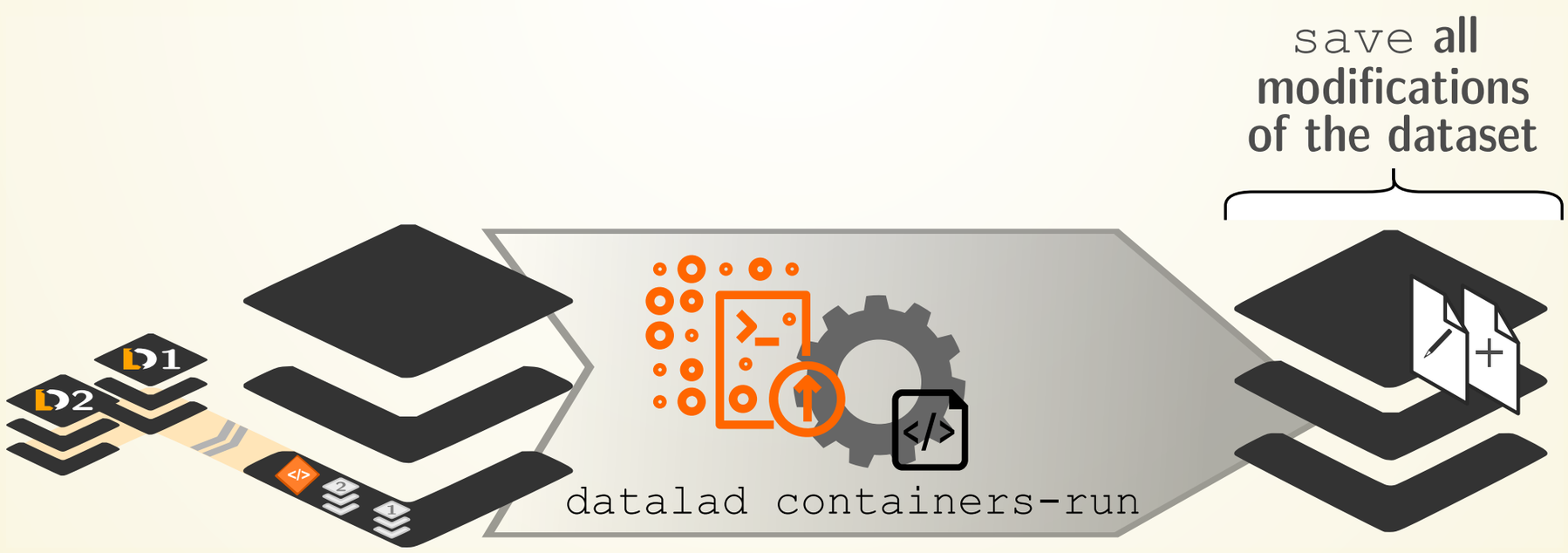
Image credit: CC BY Scriberia and The Turing Way

LEAVING A TRACE

datalad run wraps around anything expressed in a command line call and saves the dataset modifications resulting from the execution.

datalad rerun repeats captured executions. If the outcomes differ, it saves a new state of them.

datalad containers-run executes command line calls inside a tracked software container and saves the dataset modifications resulting from the execution.



DATA ANALYSIS PROVENANCE

```
$ datalad containers-run \
  --message "Time series extraction from Locus Coeruleus"
  --container-name nilearn \
  --input 'mri/*_bold.nii' \
  --output 'sub-*/LC_timeseries_run-*.csv' \
  "python3 code/extract_lc_timeseries.py"

-- Git commit --
    commit 5a7565a640ff6de67e07292a26bf272f1ee4b00e
    Author:     Adina Wagner adina.wagner@t-online.de
    AuthorDate: Mon Nov 11 16:15:08 2019 +0100
    Commit:     Adina Wagner adina.wagner@t-online.de
    CommitDate: Mon Nov 11 16:15:08 2019 +0100

    [DATALAD RUNCMD] Time series extraction from Locus Coeruleus
    === Do not change lines below ===
    {
     "cmd": "singularity exec --bind {pwd} .datalad/environments/nilearn.simg bash..",
     "dsid": "92ea1faa-632a-11e8-af29-a0369f7c647e",
     "inputs": [
      "mri/*.bold.nii.gz",
      ".datalad/environments/nilearn.simg"
     ],
     "outputs": ["sub-*/LC_timeseries_run-*.csv"],
     ...
    }
    ^^^ Do not change lines above ^^^
---
 sub-01/LC_timeseries_run-1.csv | 1 +
...
```

DATA ANALYSIS PROVENANCE

```
$ datalad containers-run \
  --message "Time series extraction from Locus Coeruleus"
  --container-name nilearn \
  --input 'mri/*_bold.nii' \
  --output 'sub-*/LC_timeseries_run-*.csv' \
  "python3 code/extract_lc_timeseries.py"

-- Git commit --
    commit 5a7565a640ff6de67e07292a26bf272f1ee4b00e
    Author:     Adina Wagner adina.wagner@t-online.de
    AuthorDate: Mon Nov 11 16:15:08 2019 +0100
    Commit:     Adina Wagner adina.wagner@t-online.de
    CommitDate: Mon Nov 11 16:15:08 2019 +0100

    [DATALAD RUNCMD] Time series extraction from Locus Coeruleus
    === Do not change lines below ===
    {
     "cmd": "singularity exec --bind {pwd} .datalad/environments/nilearn.simg bash..",
     "dsid": "92ea1faa-632a-11e8-af29-a0369f7c647e",
     "inputs": [
      "mri/*.bold.nii.gz",
      ".datalad/environments/nilearn.simg"
     ],
     "outputs": ["sub-*/LC_timeseries_run-*.csv"],
     ...
    }
    ^^^ Do not change lines above ^^^
---
 sub-01/LC_timeseries_run-1.csv | 1 +
...
```

DATA ANALYSIS PROVENANCE

```
$ datalad containers-run \
  --message "Time series extraction from Locus Coeruleus"
  --container-name nilearn \
  --input 'mri/*_bold.nii' \
  --output 'sub-*/LC_timeseries_run-*.csv' \
  "python3 code/extract_lc_timeseries.py"

-- Git commit --
    commit 5a7565a640ff6de67e07292a26bf272f1ee4b00e
    Author:     Adina Wagner adina.wagner@t-online.de
    AuthorDate: Mon Nov 11 16:15:08 2019 +0100
    Commit:     Adina Wagner adina.wagner@t-online.de
    CommitDate: Mon Nov 11 16:15:08 2019 +0100

    [DATALAD RUNCMD] Time series extraction from Locus Coeruleus
    === Do not change lines below ===
    {
     "cmd": "singularity exec --bind {pwd} .datalad/environments/nilearn.simg bash..",
     "dsid": "92ea1faa-632a-11e8-af29-a0369f7c647e",
     "inputs": [
      "mri/*.bold.nii.gz",
      ".datalad/environments/nilearn.simg"
     ],
     "outputs": ["sub-*/LC_timeseries_run-*.csv"],
     ...
    }
    ^^^ Do not change lines above ^^^
---
 sub-01/LC_timeseries_run-1.csv | 1 +
...
```

DATA ANALYSIS PROVENANCE

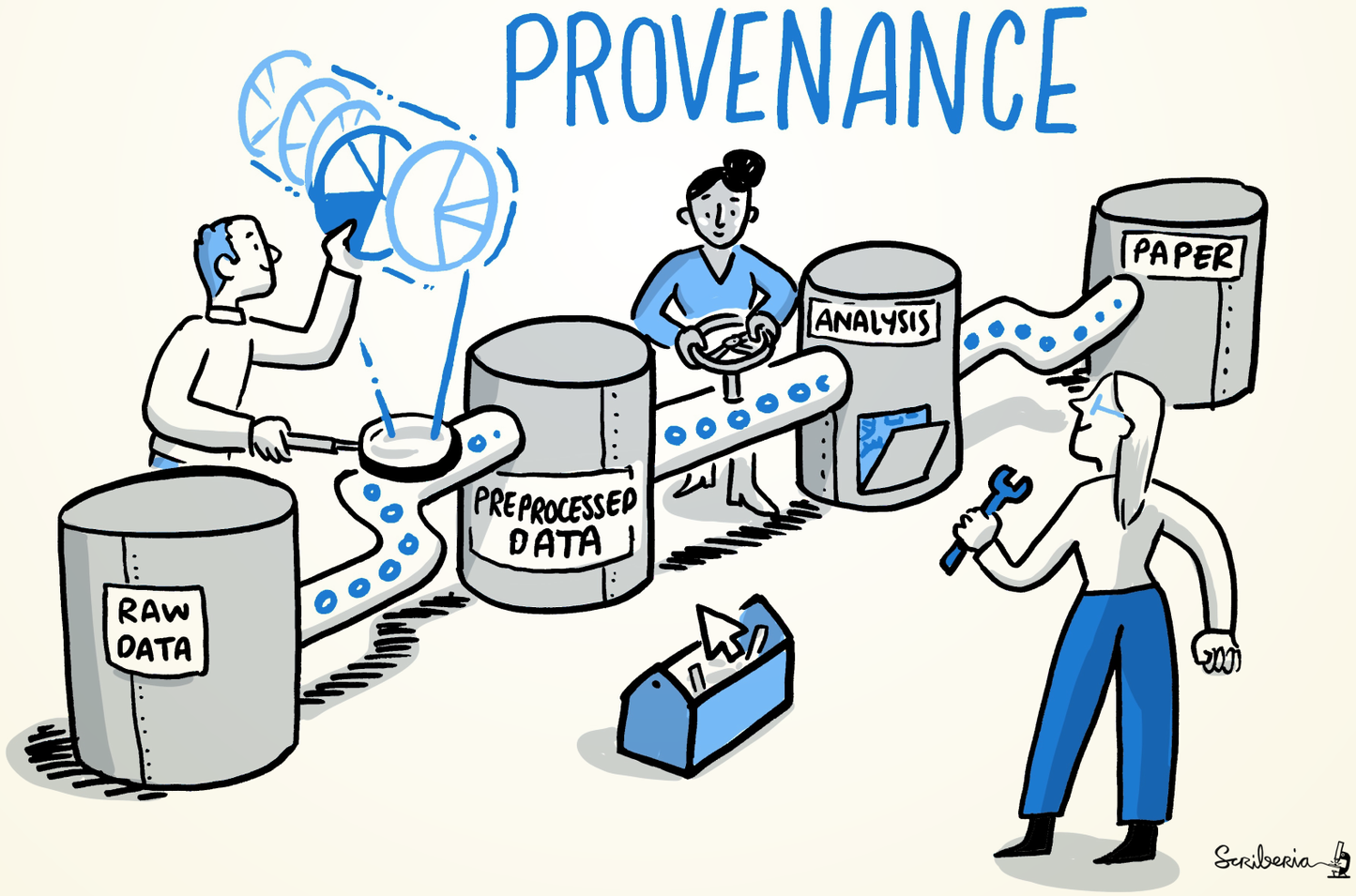
```
$ datalad rerun 5a7565a640ff6de67
[INFO   ] run commit 5a7565a640ff6de67; (Time series extraction from Locus Coeruleus)
[INFO   ] Making sure inputs are available (this may take some time)
get(ok): mri/sub-01_bold.nii (file)
get(ok): mri/sub-02_bold.nii (file)
        [...]
[INFO   ] == Command start (output follows) =====
[INFO   ] == Command exit (modification check follows) =====
add(ok): sub-01/LC_timeseries_run-*.csv(file)
add(ok): sub-02/LC_timeseries_run-*.csv (file)
        [...]
action summary:
  add (ok: 30)
  get (ok: 30)
  save (ok: 2)
  unlock (ok: 30)
```

DATA ANALYSIS PROVENANCE

```
$ datalad rerun 5a7565a640ff6de67
[INFO   ] run commit 5a7565a640ff6de67; (Time series extraction from Locus Coeruleus)
[INFO   ] Making sure inputs are available (this may take some time)
get(ok): mri/sub-01_bold.nii (file)
get(ok): mri/sub-02_bold.nii (file)
        [...]
[INFO   ] == Command start (output follows) =====
[INFO   ] == Command exit (modification check follows) =====
add(ok): sub-01/LC_timeseries_run-*.csv(file)
add(ok): sub-02/LC_timeseries_run-*.csv (file)
        [...]
action summary:
  add (ok: 30)
  get (ok: 30)
  save (ok: 2)
  unlock (ok: 30)
```

LACK OF PROVENANCE CAN BE DEVASTATING

- Data analyses typically start with data wrangling:
  - Move/Copy/Rename/Reorganize/... data
- Mistakes propagate through the complete analysis pipeline - especially those early ones are hard to find!



EXAMPLE: "LET ME JUST COPY THOSE FILES..."

- Researcher builds an analysis dataset and moves `events.tsv` files (different per subject) to the directory with functional MRI data

```
$ for sourcefile, dest in zip(glob(path_to_events),           # note: not sorted!
                              glob(path_to_fMRI_subjects)):   # note: not sorted!
    destination = path.join(dest, Path(sourcefile).name)
    shutil.move(sourcefile, destination)
```

```
eventfiles/                                analysis/                                  copy
├── sub-01                                 ├── sub-01
│   ├── events.tsv                         │   ├── bold.nii.gz
├── sub-02                                 │   └── events.tsv  # from subject 8
│   ├── events.tsv                         ├── sub-02
├── sub-03                      --->       │   ├── bold.nii.gz
│   ├── events.tsv                         │   └── events.tsv  # from subject 42
├── sub-04                                 ├── sub-01
│   ├── events.tsv                         │   ├── bold.nii.gz
[...]                                      │   └── events.tsv  # from subject 21
```

Researcher shares `analysis` with others

"I would never make such a mistake, I'm way more

- organized
- knowledgeable
- experienced

"

Everyone makes mistakes - the earlier we find them or guard against them, the better for science!

LEAVE A TRACE!

```
$ datalad run -m "Copy event files" \                                    copy
"for sub in eventfiles;
    do mv ${sub}/events.tsv analysis/${sub}/events.tsv;
done"
```

```
$ datalad copy-file ../eventfiles/sub-01/events.tsv sub-01/ -d .         copy
copy_file(ok): /data/project/coolstudy/eventfiles/events.tsv [/data/project/coolstudy/analysis/s
save(ok): /data/project/coolstudy/analysis (dataset)
action summary:
  copy_file (ok: 1)
  save (ok: 1)
```

RESEARCH DATA MANAGEMENT IS TIED TO REPRODUCIBILITY

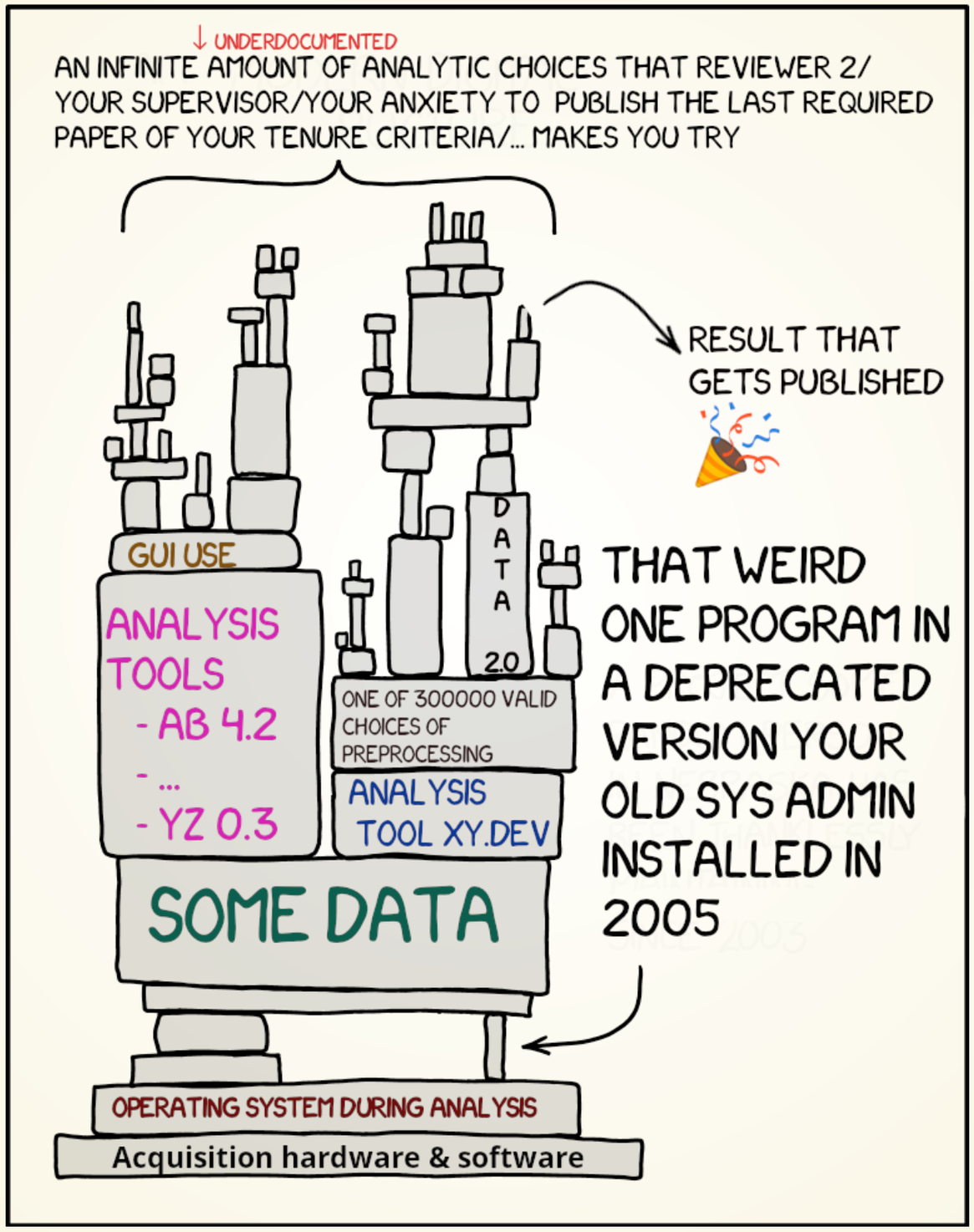


Image credit: Based on xkcd.com/2347/ (CC-BY)

Reproducibility Management in Neuroscience - Specific Issues and Solutions (DOI 10.5281/zenodo.4285927)

WHAT MAKES SCIENTIFIC WORKFLOWS SPECIAL?

Scientific building blocks are not static.

Version control beyond text

Science is build from modular units.

Nesting

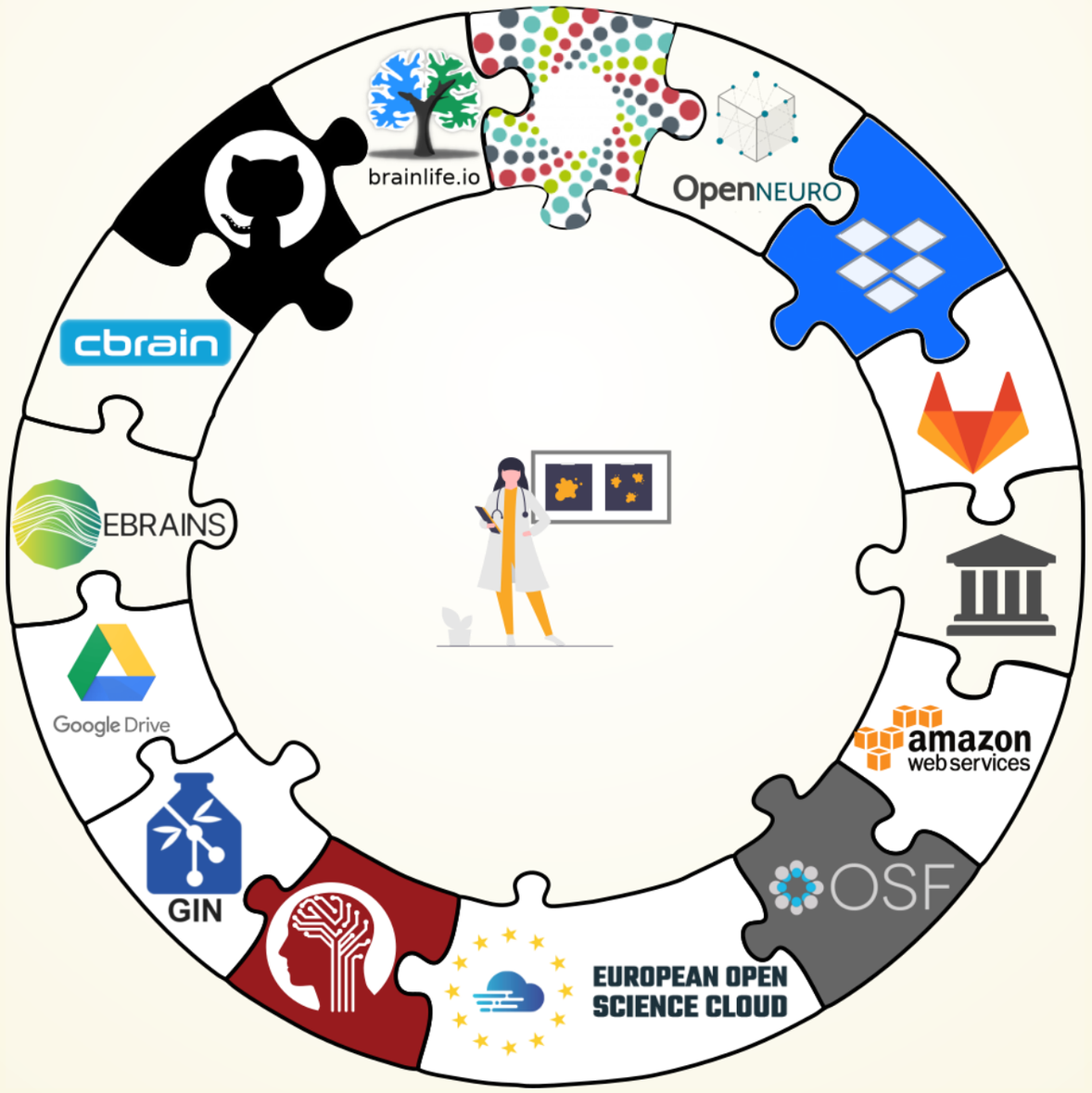
Science is exploratory, iterative, multi-stepped, and complex.

Provenance

Science is collaborative.

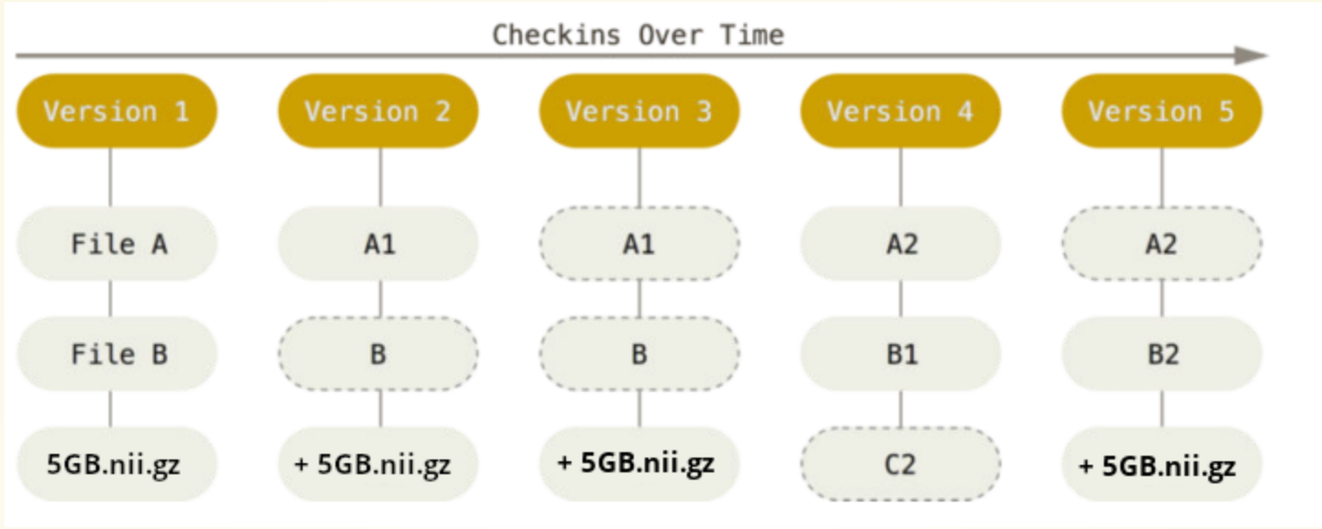
INTEROPERABILITY

- Scientific workflows can be idiosyncratic across institutions / departments / labs / any two scientists



DECENTRAL OPERATION, ALSO FOR ANNEXED FILES

Sadly, Git does not handle large files well.

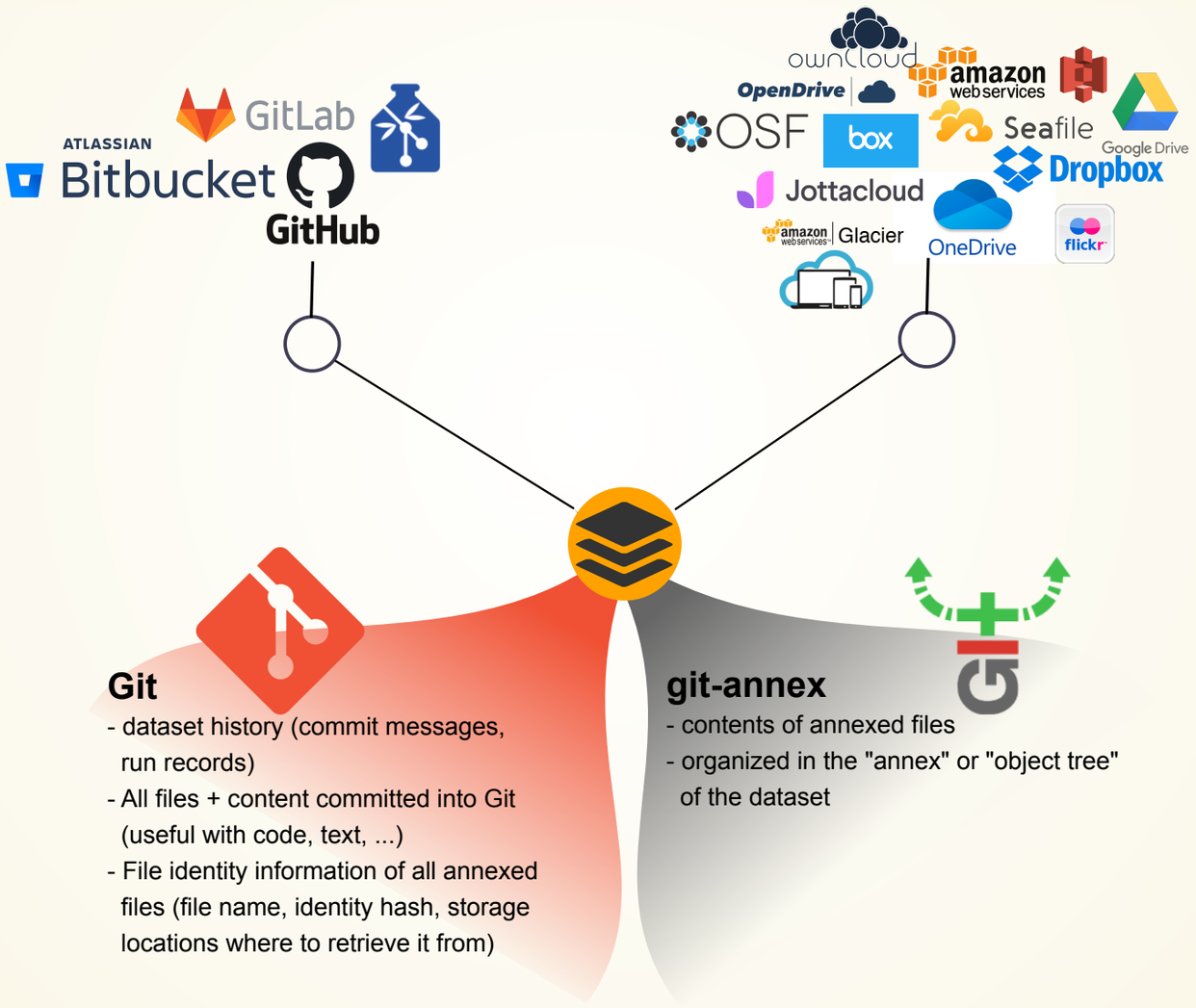


And repository hosting services refuse to handle large files:

```
adina@muninn in /tmp/myresearch on git:master
❯ git push gh-adswa master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 497.87 KiB | 161.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: error: Trace: 64a78dd41ece8e5493fe33f97397a7a90ef9c91260ba32786970dbdcf5c4e0dd
remote: error: See http://git.io/iEPt8g for more information.
remote: error: File output.dat is 500.00 MB; this exceeds GitHub's file size limit of 100.00 MB
remote: error: GH001: Large files detected. You may want to try Git Large File Storage - https://git-lfs.github.com.
To github.com:adswa/myresearch.git
 ! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'github.com:adswa/myresearch.git'
```

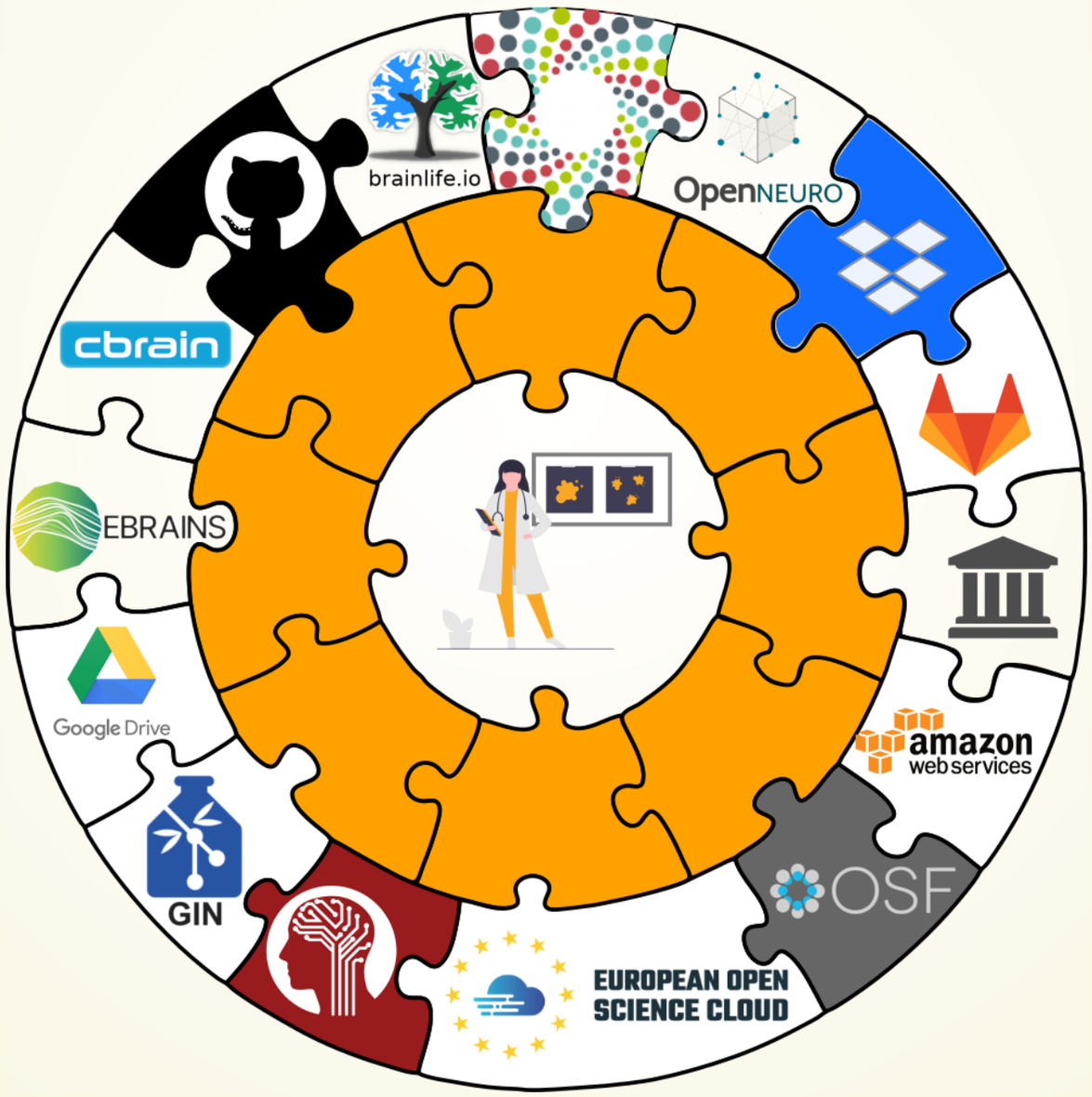
PUBLISHING DATASETS

- Most public datasets separate content in Git versus git-annex behind the scenes



INTEROPERABILITY

- DataLad is built to maximize interoperability and streamline routines across hosting and storage technology

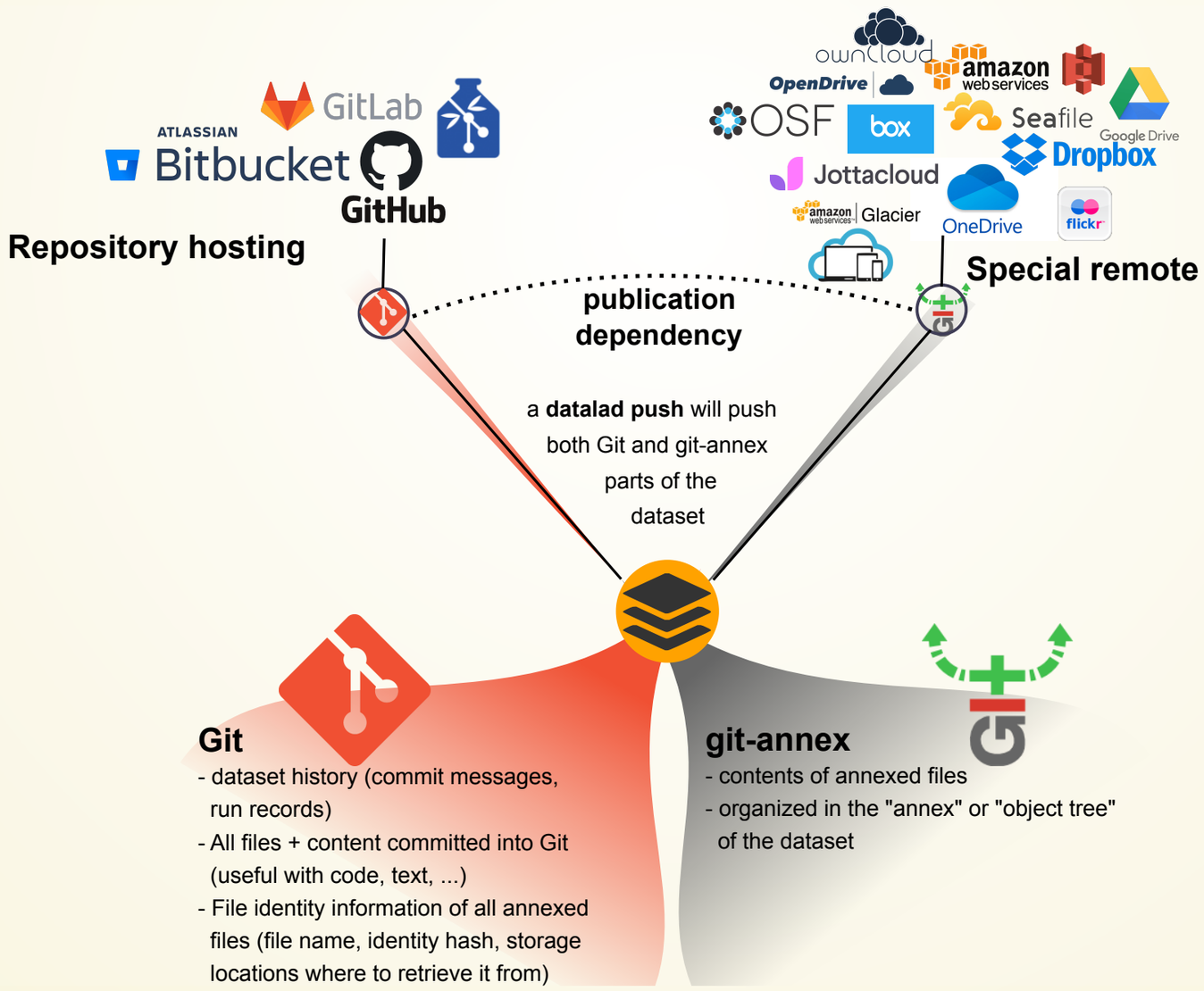


PUBLISHING DATASETS

Seamless connections:

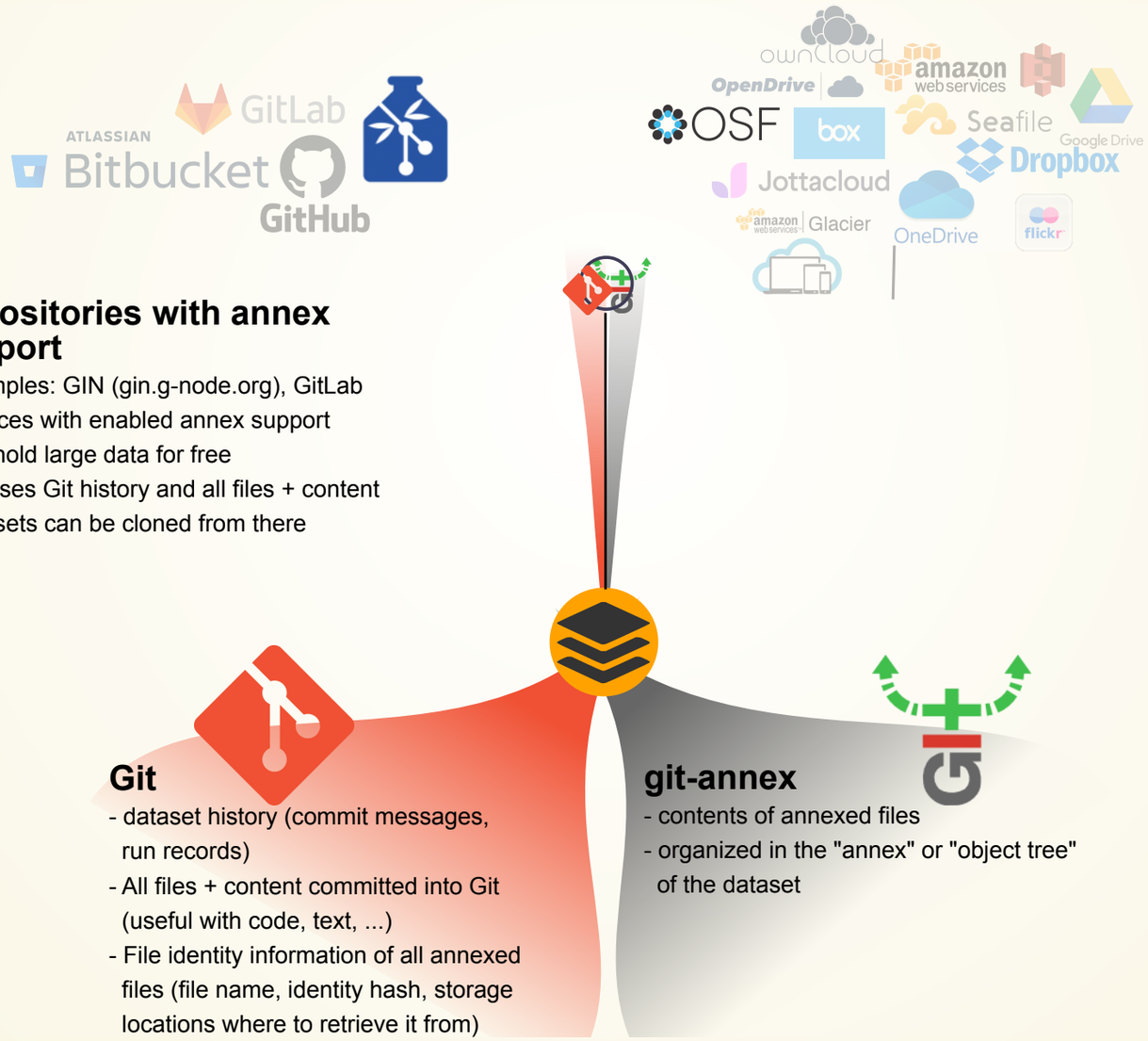
- Datasets are exposed via a private or public repository on a repository hosting service
- Data can't be stored in the latter, but can be kept in almost any third party storage
- Publication dependencies automate interactions to both places, e.g.,

```
$ git config --local remote.github.datalad-publish-depends gdrive # or
$ datalad siblings add --name origin --url git@github.com:adswa/exp-data.git --publish-depends s3
```



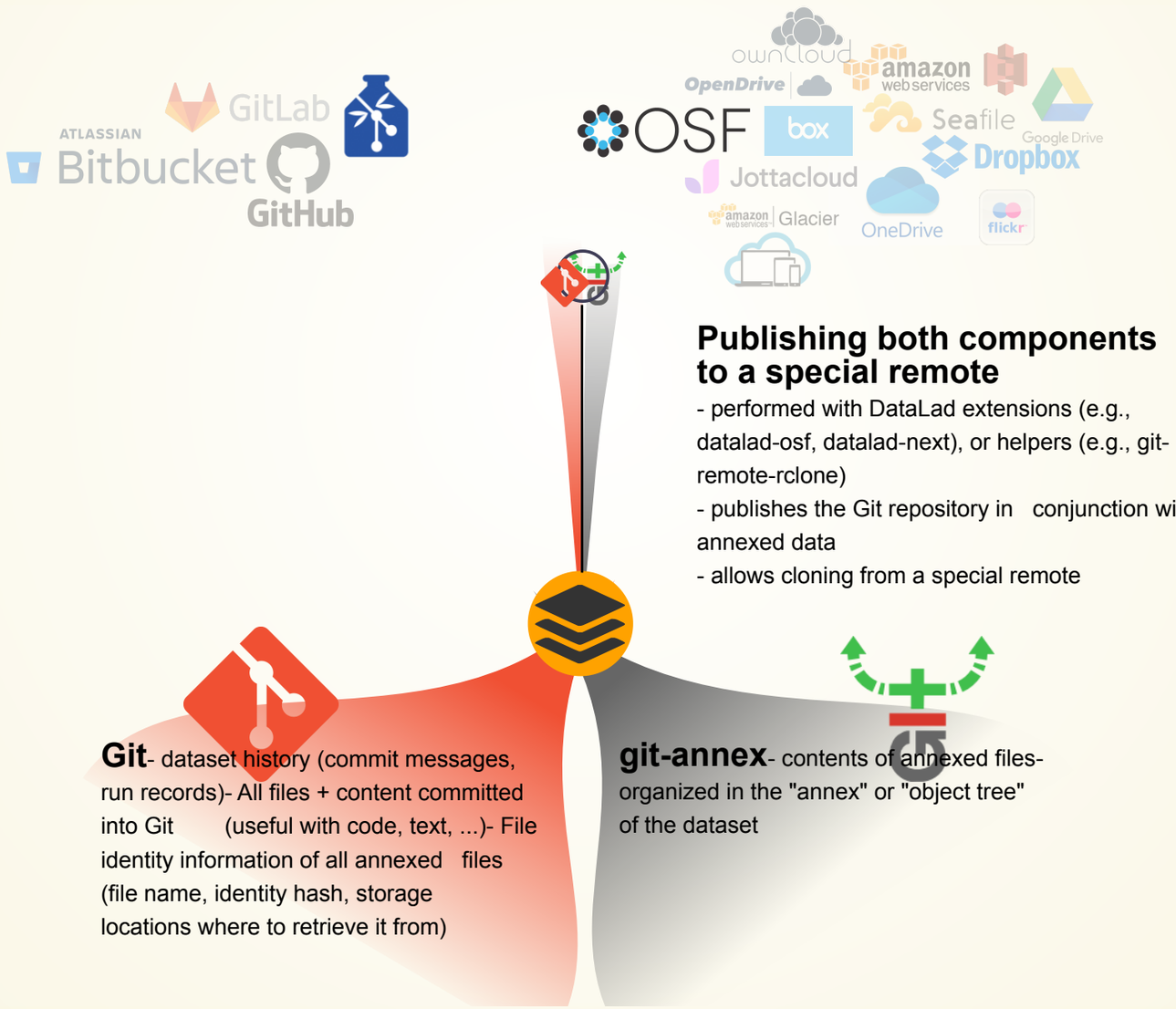
PUBLISHING DATASETS

Special case 1: repositories with annex support



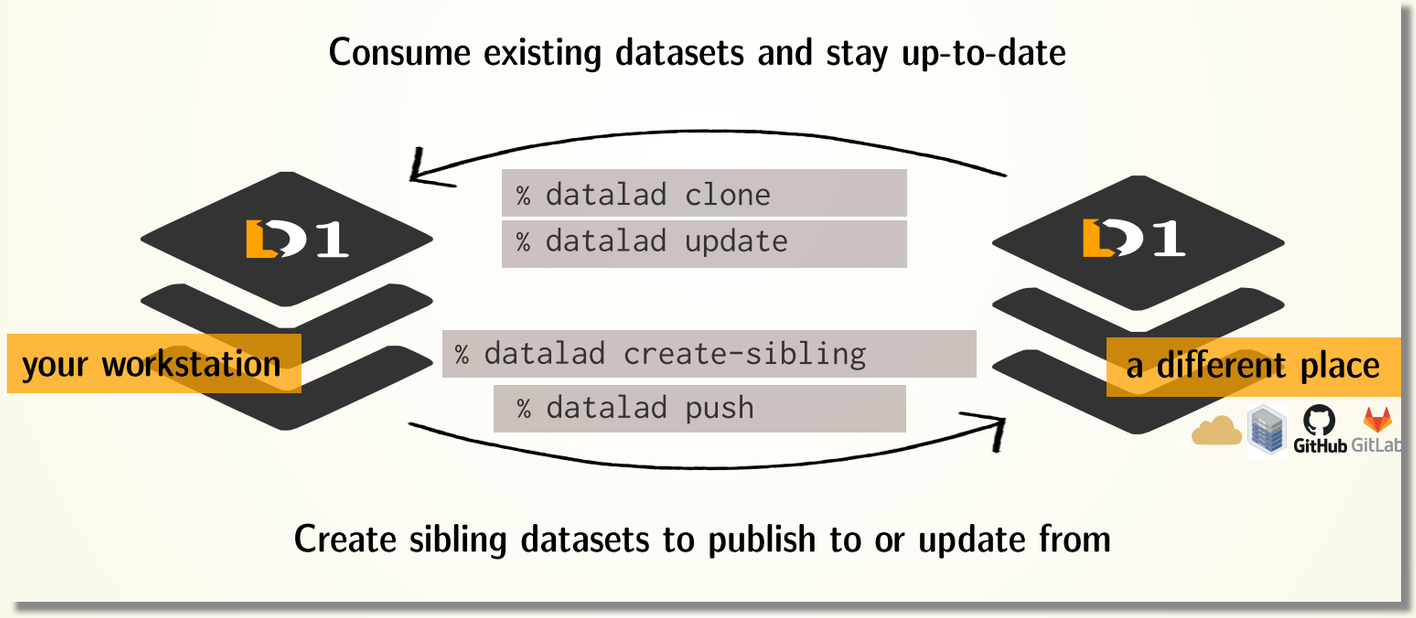
PUBLISHING DATASETS

Special case 2: Special remotes with repositories



TRANSPORT LOGISTICS

- Share data like source code



TRANSPORT LOGISTICS: LOTS OF DATA, LITTLE DISK-USAGE

- Cloned datasets are lean. "Meta data" (file names, availability) are present, but **no file content**:

```
$ datalad clone git@github.com:psychoinformatics-de/studyforrest-data-phase2.git
  install(ok): /tmp/studyforrest-data-phase2 (dataset)
$ cd studyforrest-data-phase2 && du -sh
  18M .
```

- files' contents can be retrieved on demand:

```
$ datalad get sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1_bold.nii.gz
  get(ok): /tmp/studyforrest-data-phase2/sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1
```

copy

- Have access to more data on your computer than you have disk-space:

```
# eNKI dataset (1.5TB, 34k files):
$ du -sh
1.5G  .
# HCP dataset (~200TB, >15 million files)
$ du -sh
48G .
```

copy

PLENTY OF DATA, BUT LITTLE DISK-USAGE

Drop file content that is not needed:

```
$ datalad drop sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1_bold.nii.gz
drop(ok): /tmp/studyforrest-data-phase2/sub-01/ses-movie/func/sub-01_ses-movie_task-movie_run-1_
```

When files are dropped, only "meta data" stays behind, and they can be re-obtained on demand.

```
dl.get('input/sub-01')

[really complex analysis]

dl.drop('input/sub-01')
```

(RAW) DATA MISMANAGEMENT

- Multiple large datasets are available on a compute cluster 🏞️
- Each researcher creates their own copies of data ⛰️
- Multiple different derivatives and results are computed from it 🏔️
- Data, copies of data, half-baked data transformations, results, and old versions of results are kept - undocumented 🌋

EXAMPLE: ENKI DATASET

- Raw data size: 1.5 TB
- + Back-up: 1.5 TB
- + A BIDS structured version: 1.5 TB
- + Common, minimal derivatives (fMRIprep): ~ 4.3TB
- + Some other derivatives: "Some other" x 5TB
- + Copies of it all or of subsets in home and project directories

EXAMPLE: ENKI DATASET

```
--- /data/BnB1/DATA/download_data/eNKI ----------------
                          /..
   5.2 TiB [##########] /eNKI_unzipped
   3.3 TiB [######    ] /eNKI_redownload
   3.2 TiB [######    ] /eNKI_BIDSdownload
 724.2 GiB [#         ] /eNKI_20180806
 218.8 GiB [          ] /eNKI_aus_Raw_Data
```

"CAN'T WE BUY MORE HARD DRIVES?"

NO.

DATALAD WAY

- Download the data, have a back-up
- Transform it into a DataLad dataset

```
$ datalad create -f .
$ datalad save -m "Snapshot raw data"
```

- Move it to a common location. Everyone who needs it installs it and gets required data

```
$ datalad create my_enki_analysis
$ datalad clone -d . /data/enki data
```

- Compute results with provenance capture. Drop input data and, potentially, everything that's not relevant and automatically re-computed.

WHAT MAKES SCIENTIFIC WORKFLOWS SPECIAL?

Scientific building blocks are not static.

Version control beyond text

Science is build from modular units.

Nesting

Science is exploratory, iterative, multi-stepped, and complex.

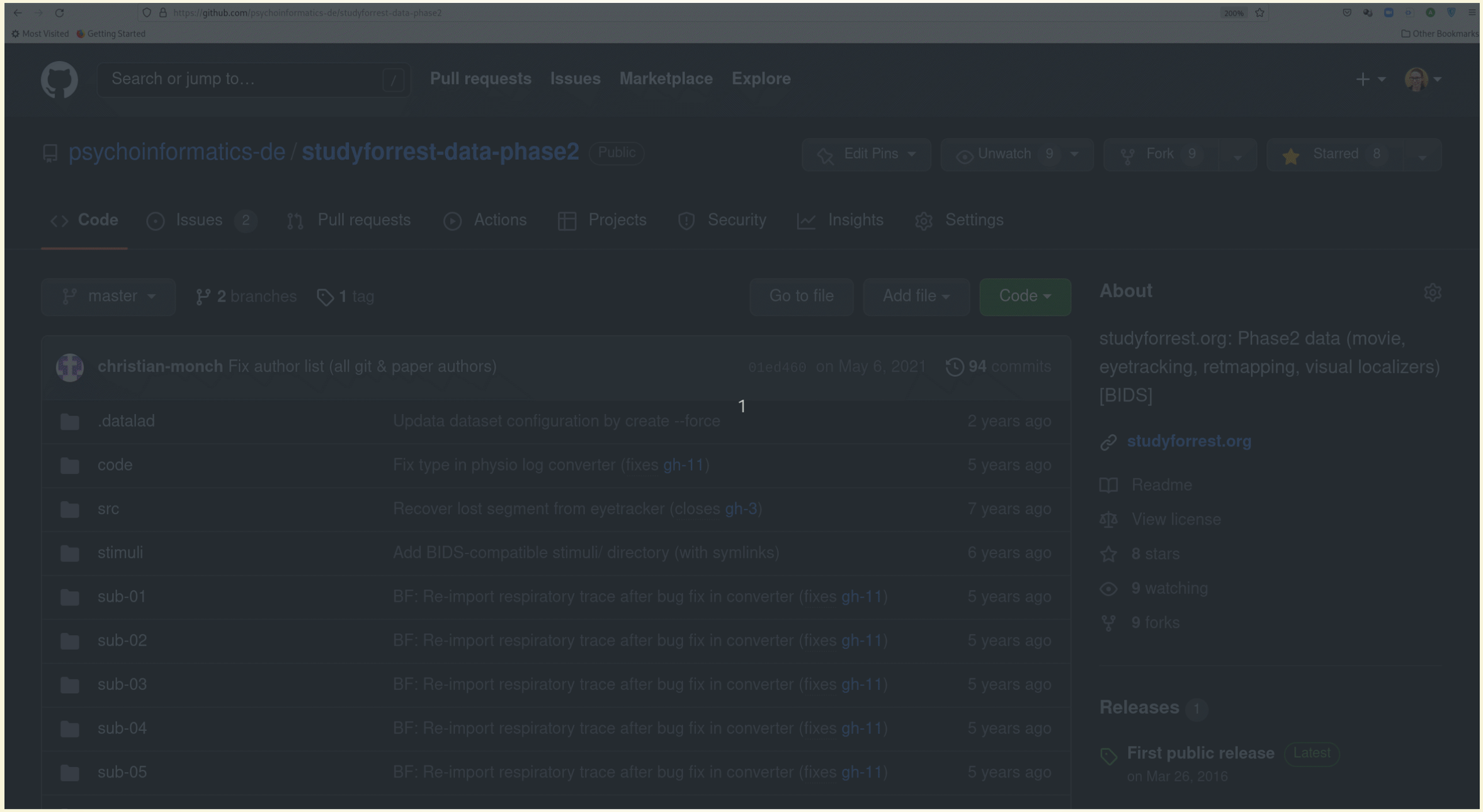
Provenance

Science is collaborative.

Transport logistics

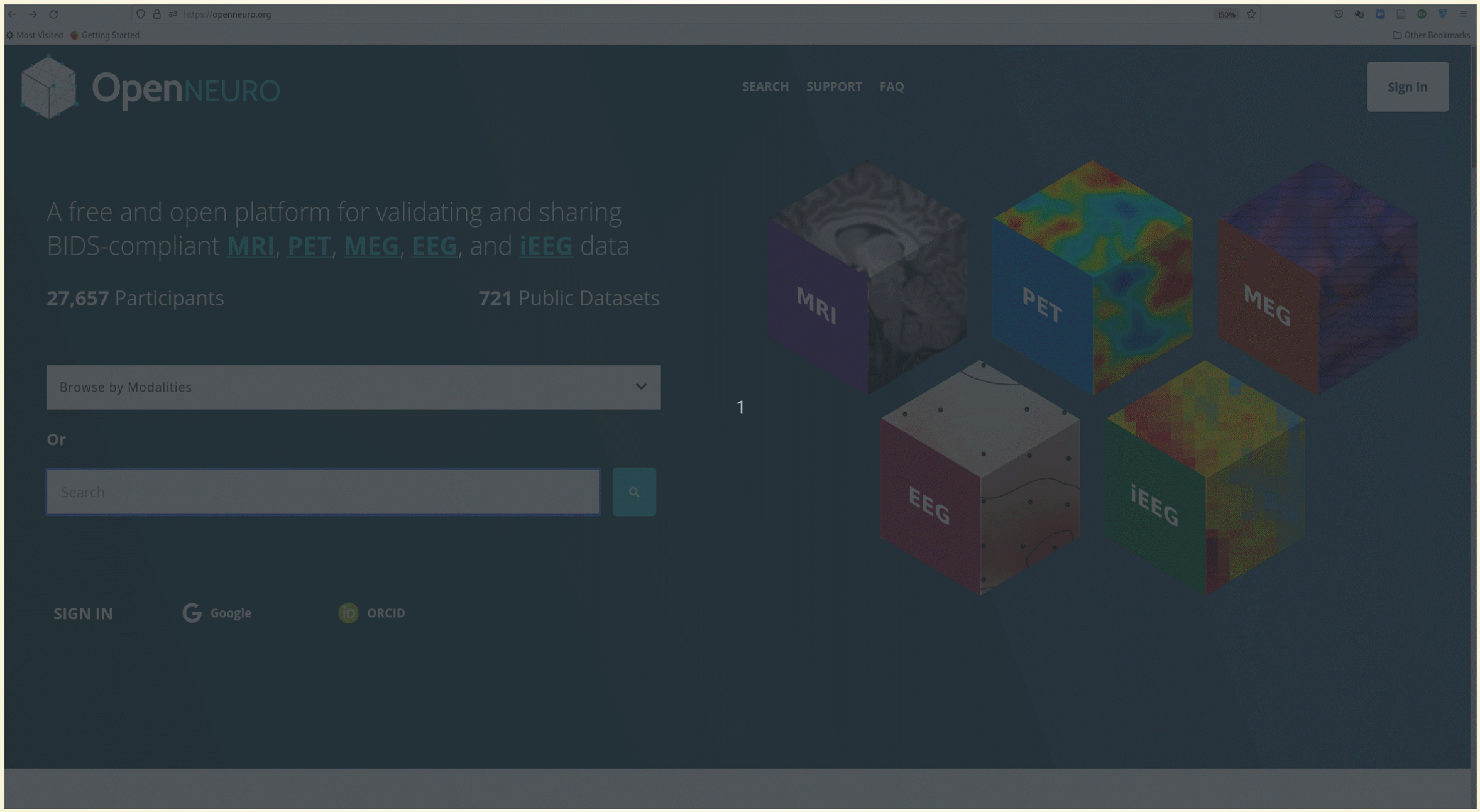
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- **Publish or consume datasets** via GitHub, GitLab, OSF, the European Open Science Cloud, or similar services



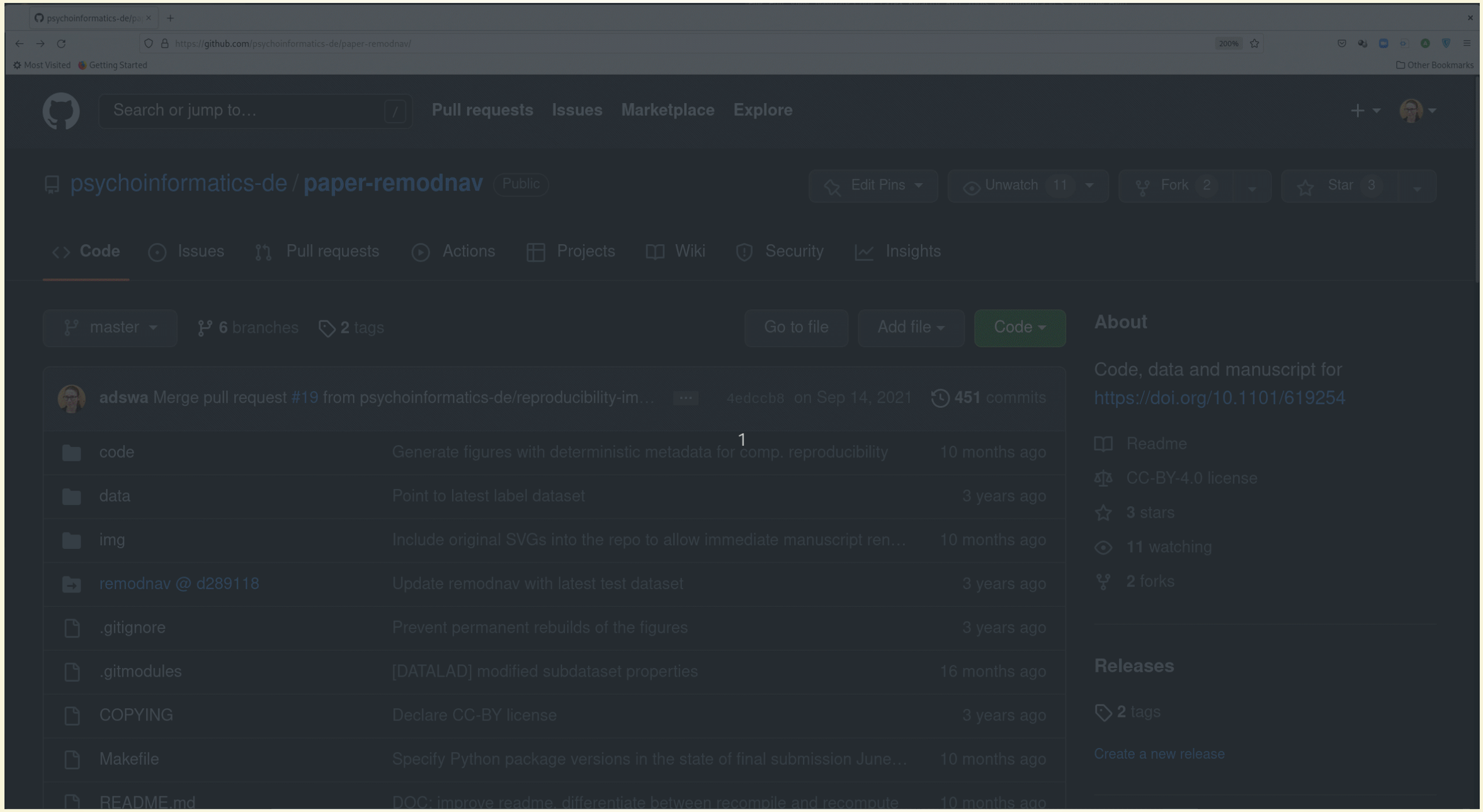
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- Behind-the-scenes **infrastructure component for data transport and versioning** (e.g., used by OpenNeuro, brainlife.io , the Canadian Open Neuroscience Platform (CONP), CBRAIN)



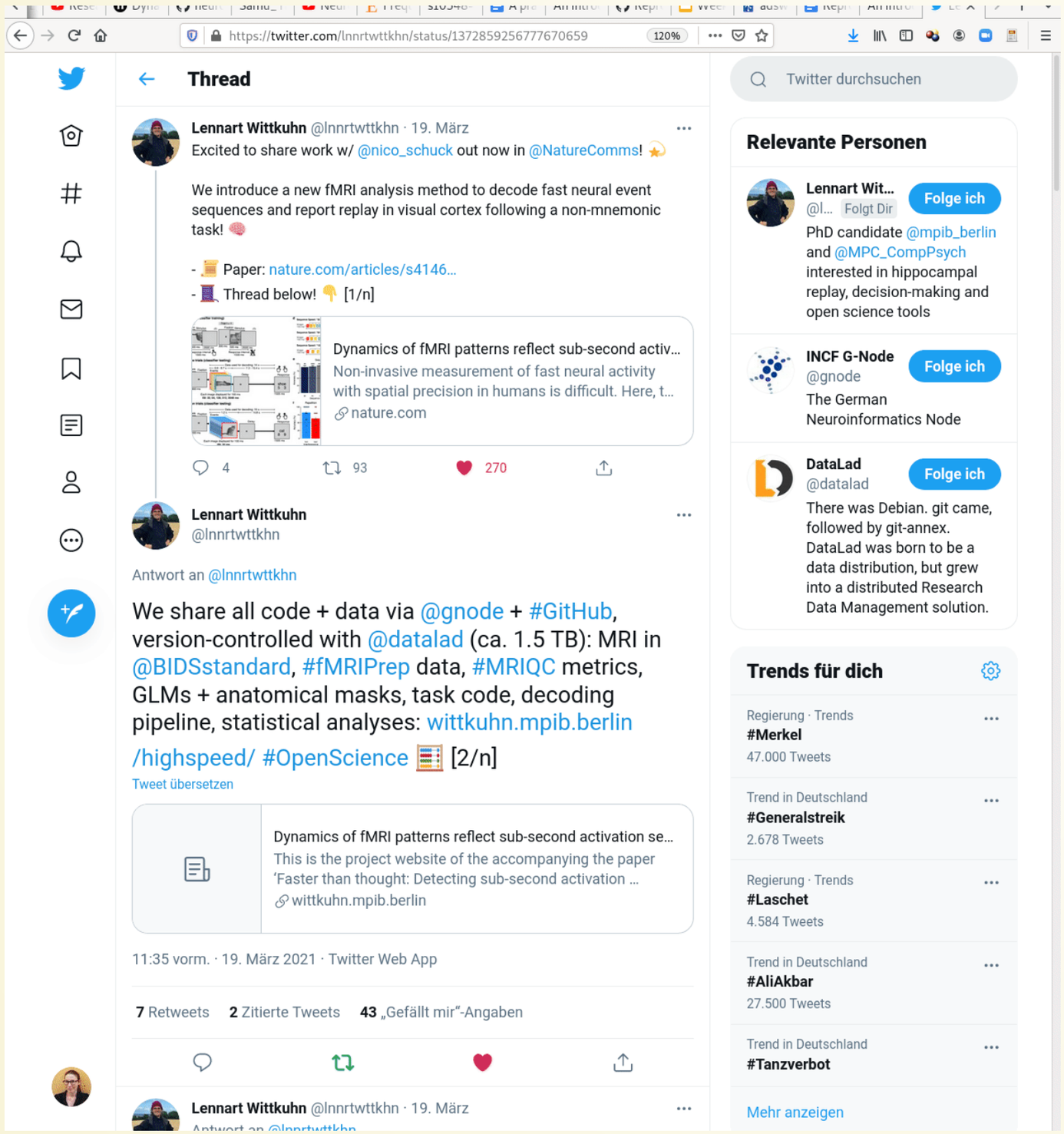
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- **Creating and sharing reproducible, open science**: Sharing data, software, code, and provenance



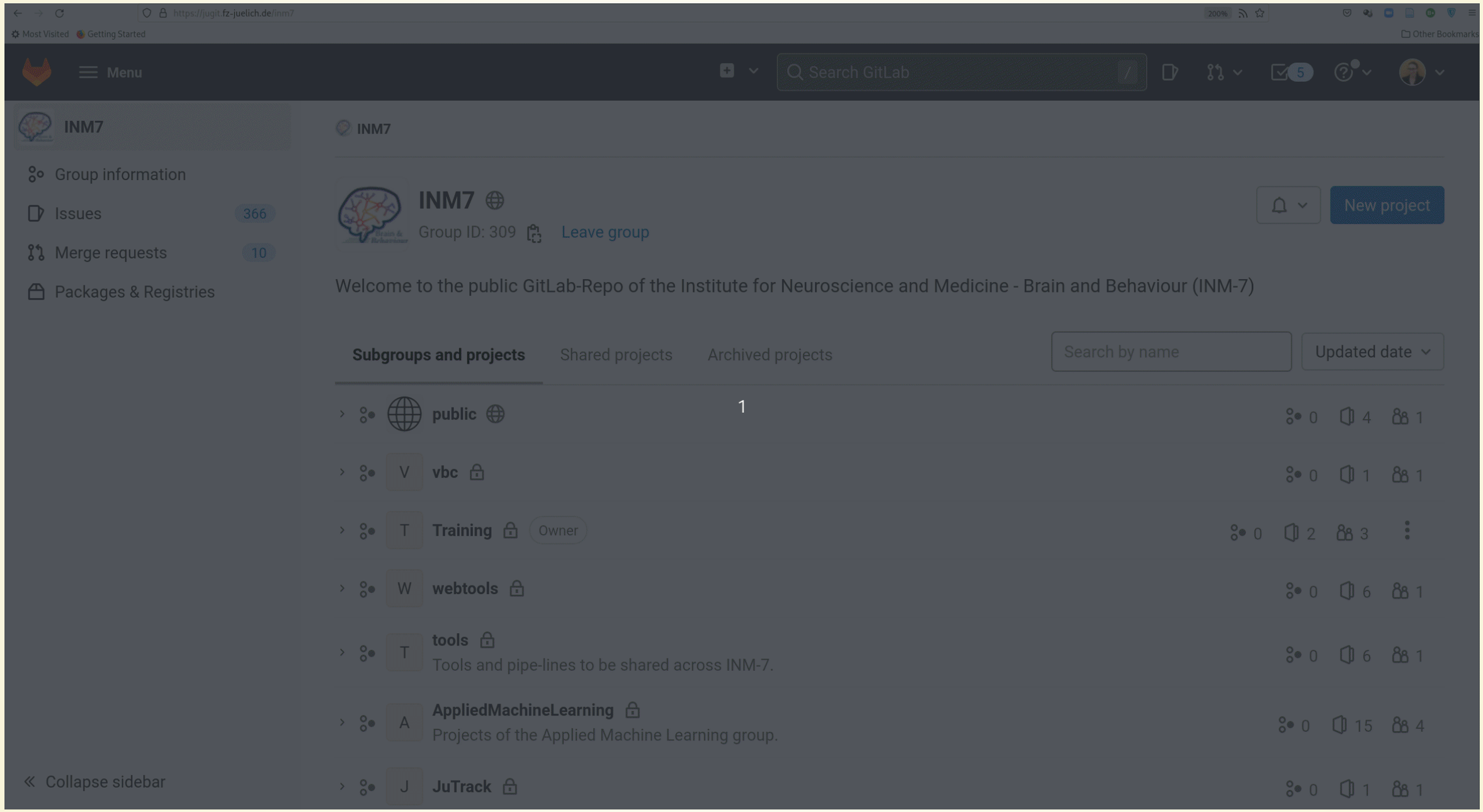
EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- **Creating and sharing reproducible, open science**: Sharing data, software, code, and provenance



EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

- **Central data management** and archival system



EXAMPLES OF WHAT DATALAD CAN BE USED FOR:

COMMAND SUMMARIES

SUMMARY - LOCAL VERSION CONTROL

`datalad create` creates an empty dataset.

Configurations (**-c yoda**, **-c text2git**) add useful structure and/or configurations.

A dataset has a history to track files and their modifications.

Explore it with Git (**git log**) or external tools (e.g., **tig**).

`datalad save` records the dataset or file state to the history.

Concise **commit messages** should summarize the change for future you and others.

`datalad status` reports the current state of the dataset.

A clean dataset status (no modifications, not untracked files) is good practice.

SUMMARY - DATASET CONSUMPTION & NESTING

`datalad clone` installs a dataset.

It can be installed "on its own": Specify the source (url, path, ...) of the dataset, and an optional **path** for it to be installed to.

Datasets can be installed as subdatasets within an existing dataset.

The **--dataset/-d** option needs a path to the root of the superdataset.

Only small files and metadata about file availability are present locally after an install.

To retrieve actual file content of annexed files, `datalad get` downloads file content on demand.

Datasets preserve their history.

The superdataset records only the version state of the subdataset.

SUMMARY - REPRODUCIBLE EXECUTION

`datalad run` records a command and its impact on the dataset.

All dataset modifications are saved - use it in a clean dataset.

Data/directories specified as `--input` are retrieved first.

Use one flag per input.

Data/directories specified as `--output` will be unlocked for modifications prior to a rerun of the command.

Its optional to specify, but helpful for recomputations.

`datalad containers-run` can be used to capture the software environment as provenance.

Its ensures computations are ran in the desired software set up. Supports Docker and Singularity containers

`datalad rerun` can automatically re-execute run-records later.

They can be identified with any commit-ish (hash, tag, range, ...)

TAKE HOME MESSAGES

Science has specific requirements that can impede efficiency and reproducibility.

- DataLad is one of many tools in an ecosystem of resources, infrastructure, and experts to assist you.
- DataLad sits on top of, and complements Git and git-annex.

Even outside of science, data deserves version control.

- It changes and evolves just like code, and exhaustive tracking lays a foundation for reproducibility.

Data management with tools like Git or DataLad can feel technical and complex.

- But effort pays off: Increased transparency, better reproducibility, easier accessibility, efficiency through automation and collaboration, streamlined procedures for synchronizing and updating your work, ...
- The biggest beneficiary of RDM? Yourself
- The second biggest beneficiary of RDM? Yourself in 6 months
- The consequence of good RDM? Better science

FURTHER RESOURCES AND STAY IN TOUCH

Reach out to to the DataLad team or contribute via

- Matrix (free, decentralized communication app, no app needed). We run a weekly Zoom office hour (Tuesday, 4pm Berlin time) from this room as well.
- The development repository on GitHub

Reach out to the (Neuro-) user community with

- A question on neurostars.org with a `datalad` tag

Find more user tutorials or workshop recordings

- On DataLad's YouTube channel
- In the DataLad Handbook
- In the DataLad RDM course
- In the Official API documentation
- In an overview of most tutorials, talks, videos at github.com/datalad/tutorials

THANKS FOR YOUR ATTENTION

Slides at DOI 10.5281/zenodo.10556597

Women neuroscientists are underrepresented in neuroscience. You can use the Repository for Women in Neuroscience to find and recommend neuroscientists for conferences, symposia or collaborations, and help making neuroscience more open & divers.